Master Thesis

Bioinformatics for DNA Repair

Supervisor Professor Tatsuya Akutsu

Department of Intelligence Science and Technology
Graduate School of Informatics
Kyoto University

J.B. Brown

February 9, 2007

Bioinformatics for DNA Repair

J.B. Brown

Abstract

DNA repair is responsible for repairing the constant damage incurred to DNA from endogenous and exogenous environmental effects. Understanding the structural and functional properties of DNA Repair improves knowledge of immunology and aids in design of new cures for illnesses. However, DNA repair is a complex system and is not fully understood.

In this thesis, we show two methods for solutions to subproblems that arise in the search for more knowledge on DNA repair. The first approach develops some structural bioinformatics techniques to predict the three dimensional structure of DNA repair proteins. The second approach utilizes machine learning techniques to identify DNA repair proteins from sequence information, opening the way to scanning new genomes or unannotated proteomes for additional DNA repair proteins that are still unknown.

This thesis provides an overview of the problem with a review of related literature, a discussion explaining the prerequisite knowledge to understand its contents, chapters for the two aforementioned subproblems along with experimental results, and a concluding elaboration of research topics that build on the work completed and shown in this thesis.

DNA 修復におけるバイオインフォマティクス

ジェービー・ブラウン

内容梗概

DNA 修復というプロセスとは、紫外線や新棟代謝から出現する酸素などの細胞内外の影響によって損傷される DNA を修復することである。DNA 修復を構成的及び機能的に理解できれば免疫学の知識が増し、新しい薬が作れるようになり、治療も可能となる。しかしながら DNA 修復は多数のメカニズムが関係しているため、完全に把握されていない現象である。

この論文では、DNA 修復に関する二つの問題を研究する。一つ目は、DNA 修復タンパク質の立体構造を予測する問題でこれに対していくつかのアルゴリズムを提案する。タンパク質の機能を予測するために精度の高い立体構造情報が要り、この情報をアルゴリズムで予測する。二つ目は、DNA 修復タンパク質を認識する問題である。このため、機械学習という分野の技術を利用し、配列情報などから、あるタンパク質が DNA 修復に関係するか否かを判定する。この技術を利用すれば新しいゲノムや機能未知なタンパク質をスキャンして新しい DNA 修復タンパク質が発見できる。認識された DNA 修復タンパク質をさらにいくつかの種類に分類する。

論文の構成は次のとおりである。まず DNA 修復という問題を説明する上で必要となる関連文献を紹介する。そして、この論文の内容を理解するための基本知識を提示する。さらに上記の研究内容をより詳しく説明して実験結果を示してから結果について考察を行う。そして、この論文の内容を今後の研究でいかに利用するかについて議論する。

Bioinformatics for DNA Repair

Contents

Chapter 1 Introduction

In this chapter, we present an overview of what DNA repair is and why it is important, review some of the literature and existing knowledge on DNA repair, and finally discuss what this thesis sets out to accomplish in the field of DNA repair.

1.1 What is DNA Repair?

Our world is a woven fabric of thousands of living organisms, from small microbes that can only be seen under the microscope to large mammals the size of a school bus. Yet despite such a wide range in size and sophistication, all organisms have a number of fundamental biological processes in common. In particular, modern microbiology and genetics teaches us that all living organisms are capable of forming structured organizations of their constituent matter, better known as cells, and are capable of maintaining, using, and passing on to a new generation a series of instructions for their life spans, modernly described as DNA. The emergence of DNA from RNA [2] (see Figure 1) as the blueprint for life brought about a considerable improvement in stability of the genetic material that constitutes the living world.

In spite of the improvement, the world around us constantly submits DNA to a myriad of destructive factors. These factors have the ability to either destroy or modify the stability of life's blueprint. In all but rare situations, these errors result in apoptosis, or cell death.

For example, normal human cells lose thousands of base residues every day through a process called hydrolysis [20]. Another example is normal metabolism, which produces oxygen free radicals as a by-product. Despite common knowledge that oxygen is necessary for human beings, these free radicals are actually detrimental to the consistency of DNA [20]. In addition to these endogenous processes, there are exogenous processes that damage DNA as well, such as hydrocarbons found in cigarette smoke [25] or ultraviolet (UV) radiation absorbed by prolonged exposure to the sun [20]. Absorption of X-rays or gamma rays can break either one or both of the strands holding DNA together [25] (see the

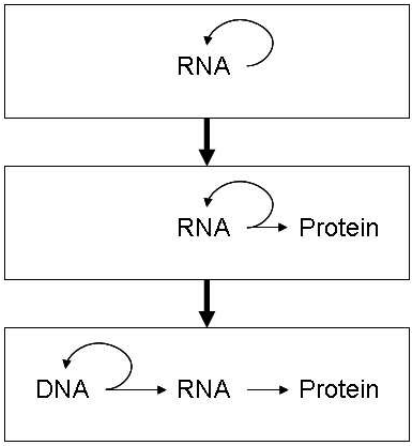


Figure 1: Primitive RNA served as both the information carrier and reaction catalyzer. As time evolved, proteins developed to assume the functional roles done by RNA up to that point. Even later in time, DNA is thought to have emerged because of its stability in preserving information. Presently, RNA still maintains its functional and informational roles, but in a reduced capacity relative to its origin.

bottom half of Figure 3). Chemotherapy chemicals that are used to treat cancer can cause DNA crosslinks which are detrimental to cell health [25, 30].

Without some mechanism to withstand this damage, the homestasis and evolution of living beings would be nearly impossible. Emerging in response to these destructive factors, nature has developed a number of mechanisms that repair DNA to its normal state. For the purposes of this thesis, we define DNA repair to be *the collection of proteins and associated mechnisms by which normal DNA sequence information is restored after some perturbation.* These DNA repair mechanisms have been classified into many categories, with major categories such as excision repair or mismatch repair. A more complete list of excision mechanisms is given in Table 1. Several DNA repair mechanisms are illustrated in Figures 2 and 3.

Table 1: An assortment of DNA repair mechanisms, organized by major category in accordance with the listing in [20].

Repair Category	Repair Mechanisms
Excision	Base excision repair
	Nucleotide excision repair
	Transcription-coupled nucleotide excision repair
	Alternative excision repair
	Mismatch repair
Strand Break	Single-strand break repair
	Double-strand break repair
Direct Reversal	Reversal of base damage

1.2 Relevant Literature

Research on how and what damages DNA, as well as what genes (and hence proteins) are responsible for repair has been ongoing for many decades. As a result, we know that there are at least 130 genes involved in DNA Repair [25], and we also have a fairly good understanding of the major mechanisms of DNA repair [20, 30]. For example, a series of articles appearing in *Nature* have shown how the UvrA, UvrB, and UvrC genes are active in *E. Coli* excision repair [32], while the XPA, XPB, and XPC genes are just three products involved in transcription-coupled nucleotide excision repair, a repair mechanism whose failure has been linked to xeroderma pigmentosum (XP), Cockayne's syndrome (CS), and trichothiodystrophy (TTD) [21].

Every year new knowledge on DNA is becoming available, giving researchers the sense that the field still has much to be explored. In 2005, Kimball reported that there were at least 11 DNA polymerases encoded by our genes [25], though only a year later Linn gave a talk in which he discussed the presence of at least 17 DNA polymerases [27]. So while each of these repair categories has been observed and characterized in laboratories, there still remain many open questions in regard to the complete and complex process known as DNA Repair.

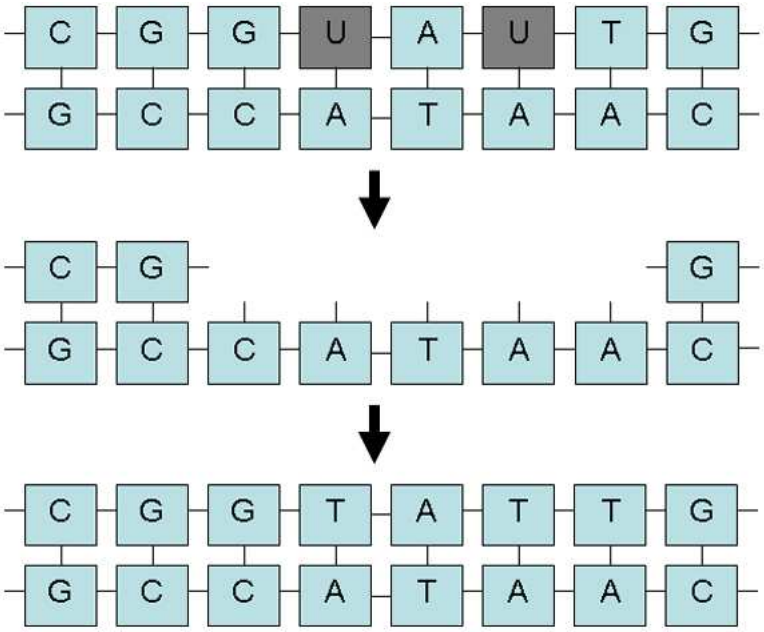


Figure 2: The DNA excision repair system replaces a damaged or incorrect nucleotide as well as the area around it. In this example, it replaces the mistaken nucleotides (U) and the area around them with the correct nucleotides (T).

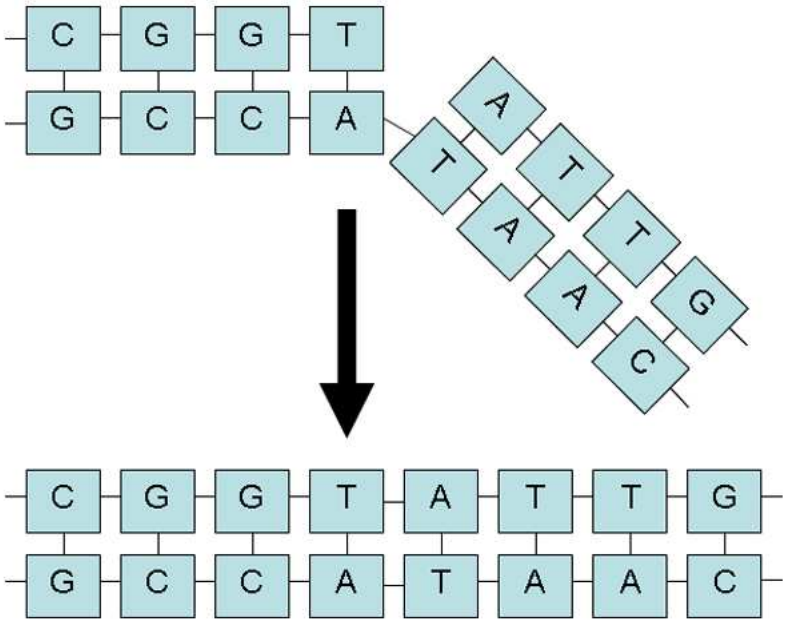


Figure 3: After UV radiation breaks one of the strands of the DNA backbone, the strand break repair system seals the created gap to restore DNA's stable structure.

1.3 Purpose of this Thesis

In the mid-1970's microbiology and genetic research began to take advantage of the memory capacity and speed of computers. Shortly thereafter, technologies such as microarrays and DNA chips created an explosion in available genetic data, facilitated by the continual expansion in capacity of computer hard disks. At the beginning of the 21st century, the field of bioinformatics (also known as computational biology), which is the mathematical analysis and subsequent experimentation on genetic data using computational means, has rapidly emerged as a relevant field for biological, genetic, and medical advances. At this point, it is appropriate to introduce the vantage point and aims of this thesis.

As complete genomes are continually being sequenced and the field of bioinformatics has produced a wealth of sequence information and annotation, it is advantageous to search for additional DNA Repair knowledge using computational techniques. As mentioned, there are still many questions that remain unsolved in DNA repair, and introducing computational approaches that can not only verify existing knowledge but also uncover new knowledge in a systematic matter is a considerable tool to further our knowledge on the topic. It is, of course, necessary to verify the knowledge uncovered through analytical and computational means, but the unquestionable advantage of such an approach is that it reduces the amount of manual labor as well as trial and error to search for new knowledge. For example, if we are looking for which combination of six genes are responsible for a particular protein's production, then there are $2^6 = 64$ possible combinations to search through. Trial and error of these combinations in a laboratory may require days, weeks, or months. However, if we have established an accurate computational model by which we can test each of the possibilities, we may be able to search through the entire set of combinations in a matter of seconds.

It is in this spirit that this thesis looks to build computational models in search of new knowledge for DNA repair. Successful research in DNA repair opens new paths to knowledge in immunology, pharmacology, biology, genetics, or medicine. Hence, in this thesis, we show solutions to solve two problems, DNA repair protein structure prediction and DNA repair protein identification,

in the larger quest for more complete knowledge on DNA Repair.

We begin with a preliminary discussion of the necessary biological and mathematical knowledge as well as the information processing techniques needed to solve these two problems. With the required background in hand, the side chain packing problem, a structural bioinformatics problem, is addressed with the goal of predicting the three dimensional structure of DNA repair proteins. Our focus then shifts to the large-scale identification of DNA repair proteins, utilizing a highly successful machine learning approach. We follow up with concluding remarks, including a considerable discussion of future works that can be launched as a result of the steps taken in this thesis.

Chapter 2 Background

In this chapter, we discuss each type of prerequisite knowledge required to understand the methods which are utilized in this thesis. Since DNA repair is a considerably biological topic, we provide a primer in basic microbiology. We continue to a discussion on graphs, as they are the cornerstone of the work done in Chapter 3. Finally, Chapter 4 makes heavy use of machine learning and kernel methods, so to ensure that the biological community can also understand the material here, we cover the concepts of machine learning at a fairly abstract level.

2.1 Biological Background

It is common knowledge in the 21st that DNA is the blueprint to life. However, how that blueprint is used and what happens as a result of DNA is not as commonly understood. As explained in Section 1.1 and Figure 1, there are at least three major molecules that govern life. In this section, we discuss the interplay of those three molecules and how DNA repair adds to nature's canvas of molecules and interactions.

2.1.1 The Central Dogma of Molecular Biology

After Francis Crick and James Watson determined the structure of DNA in 1953 [17], three of the major and critical components in life were identified: *DNA*, *RNA*, and *protein*. So that this thesis is readable by the general public as well, we explain each of these components in brief detail.

The constituents of DNA are known to be nucleotides based on the four bases adenine, cytosine, guanine, and thymine (abbreviated A,C,G, and T). In all living organisms, one will find only these four bases existing in DNA. Watson and Crick uncovered the fact that DNA is a double-stranded molecule, and that each strand is complementary to the other with a simple pairing rule. By being complementary, a given or missing strand's information can be deduced if we know the other strand that the given or missing strand partners with. Since the pairings are normally C-G and A-T, then, for example, from a single strand of DNA that has the sequence $s_{one} = GCCATAAC$, we know that the other

strand must have the sequence $s_{two} = CGGTATTG$. This sequence pair is illustrated at the bottom of Figures 2 and 3 in Chapter 1.

RNA, mentioned in Chapter 1 as a molecule which originally was both the information carrier and the reaction catalyst, maintains a similar pairing rule to DNA, but with one change: the thymine (T) nucleotide is replaced by the uracil (U) nucleotide. This gives us the pairings C-G and A-U for RNA. The reason why the uracil residues in Figure 2 are replaced should now be clear - they do not belong in a DNA molecule. Excision repair, a type of DNA repair discussed in Chapter 1, has the ability to remove the uracil residues that can found in a DNA molecule because of a DNA replication proofread failure.

Proteins are molecules in the body which serve many structural and functional roles, performing as antibodies, toxins, hormones, antifreeze molecules, elastic fibers, and ropes [2]. They are made up of chains of amino acids, much as DNA is made of up long chains of nucleotides. In all but extremely rare situations, there are exactly 20 amino acids which constitute modern proteins. The amino acids are listed in [2], and are represented by the letters of the alphabet except for the letters B, J, O, U, X, and Z.

In 1961, C.B. Anfinsen showed that a protein's three dimensional structure in nature is largely predetermined by its sequence of amino acids [3]. Any given protein structure has a specific role as a result of that structure [2]. Thus, if we want to understand the role that a particular protein has in a structural or functional process (say, muscle tissue or blood clotting), we must understand its structure, which is in turn given through its amino acid sequence.

With this knowledge, we ask the final question of this section: how are DNA, RNA, and proteins connected? The key is that sequences in DNA form codons, triplets of nucleotides, which correspond to particular amino acids. En route to being the signal to building an amino acid, each codon in DNA is transcribed into RNA. So RNA provides the link between DNA and proteins, and in this context, the evolution of RNA in Chapter 1 (Figure 1) makes sense. How DNA, RNA, and proteins come together to form a more complete system is known as the *central dogma of molecular biology*, and is illustrated in Figure 4.

This thesis makes extensive use of the central dogma of molecular biol-

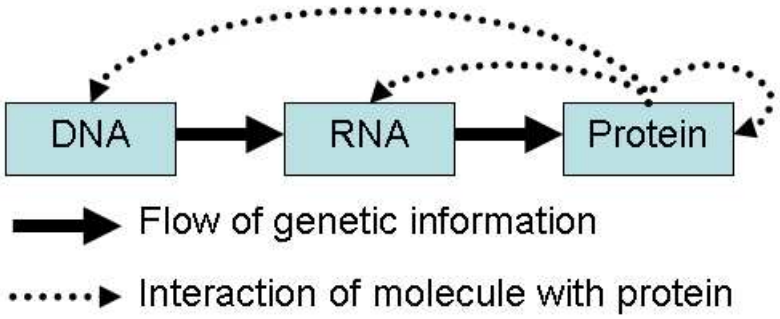


Figure 4: The central dogma of molecular biology. The blueprint for life is stored in DNA and transcribed to RNA, which is then translated into a sequence of amino acids for building proteins. Proteins interact with DNA, RNA, and other proteins at each step.

ogy. In Chapter 3, we will combine sequence information and graph theory (to be explained in Section 2.2) to develop an approach for predicting the three-dimensional structure (final conformation) of DNA repair proteins. In Chapter 4, we will extract features from the sequences of DNA repair proteins to develop predictive techniques for finding novel DNA repair proteins.

2.1.2 DNA Repair Protein Motifs

It was discussed in Chapter 1 that DNA repair comes in many forms, such as the two major distinctions of direct reversal and excision repair. Here, we explain a little more about each of the types of repair as well as detail several types of repair that were not given in Chapter 1.

The information-determining material of DNA, that is the bases adenine, cytosine, guanine, and thymine, readily interact with water and oxygen at normal temperatures around 37°C [20]. When that interaction takes place, it changes the chemical properties of the base, essentially turning it into a different base, which means that the encoding information is no longer correct. *Direct reversal* proteins are capable of detecting this mistake and fixing it. However, in the interaction to reverse the damage, the protein must "sacrifice" itself, and hence cannot repair another altered base. As a result, direct reversal is sometimes said to be ineffective [25].

There are times when DNA cannot be simply reversed as explained in the

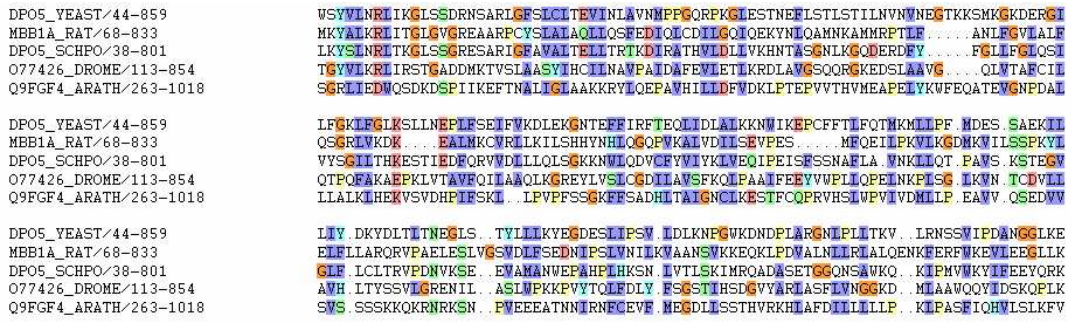


Figure 5: The Pfam multiple sequence alignment output for DNA polymerase phi. Colored individual letters indicate where alignment occurs and the types of amino acids involved. The alignment is shown for the first 240 amino acids of the five seed organisms, divided into 80 amino acids per row.

preceding paragraph. In [20], this is explained with the analogy that a knot in a string can sometimes be undone, but sometimes the knot must simply be cut out and new strings re-attached. Excision repair follows the analogy of cutting the string and replacing the necessary parts. The concept of excision repair was shown in Figure 2 in Section 1.1. When a mistake in DNA information occurs and direct reversal is not possible, it is easier and more "economical" for the repair system to simply cut out the mistake and its immediate nucleotide neighbors (similar to cutting a knot and a small length of string leading to the knot) rather than take all of the steps to undo the mistake.

Variations on excision repair are abundant. Proteins that verify only transcriptionally active genes, that is genes that are being processed according to the central dogma of molecular biology, are classified as *transcription-coupled excision repair* (TC-NER) proteins. Another excision repair type involves cutting only one part of the DNA backbone in the mistaken region, and is appropriately termed *alternative excision repair* (AER) because of its variation on the typical dual incision of the DNA backbone. Many more types of excision repair exist, and the reader is referred back to Table 1.

Repair mechanisms other than those listed in Table 1 also exist. The DNA repair and replication machinery has the ability to use a single strand of normal DNA and fold it back over itself at a replication fork to create a correct partner

strand that will replace a damaged strand already in existence [20]. This is called *excision fork fold back repair*. The repair system may allow a certain amount of mutation to occur if it cannot repair the errors fast enough or it determines that the damaged DNA is in a transcriptionally silent region. This is the basis of *mutagenesis*, to which the reader is referred to [20] for a thorough discussion.

Amongst all of these types of DNA repair, we ask the question, "Are there any general patterns in DNA repair proteins?" Since proteins are comprised of sequences of amino acids, we would like to look at many sequences at once to examine if there are any general patterns in DNA repair proteins. Using the Pfam database [9], multiple sequence alignments can be done for DNA repair proteins.

We mentioned in Section 1.2 that in 2005 there were 11 known DNA polymerases, with that number further expanded to 17 in 2006 [27]. Amazingly, in 2007, a query for "DNA repair polymerase" in the Pfam database led us to the *DNA polymerase phi* protein. This is significant because phi is the 21st letter of the greek alphabet, inferring that the uncovering of DNA polymerases is still a current research topic and furthermore that there is still more to be learned in DNA repair. As a result, we requested a multiple alignment of the DNA polymerase phi found in five core organisms, which is shown in Figure 5. While DNA polymerase phi is considerably new and has not been identified in many organisms, the UV radiation repair proteins uvrB and uvrC, first mentioned in Section 1.2, have been sequenced in many more organisms. Similar to DNA polymerase phi, an alignment of uvrB and uvrC proteins in many organisms, shown in Figure 6 reveals that certain amino acids are prominent in particular sequential positions.

Multiple alignments of DNA repair proteins open the path to using domain-specific knowledge to develop techniques for automatically identifying novel DNA repair proteins. In Chapter 4, we will specifically use the sequence information of DNA repair proteins to teach a machine how to automatically identify other DNA repair proteins in a set. Additionally, the DNA repair protein motifs discussed here and shown in Figures 5 and 6 can be considered for further

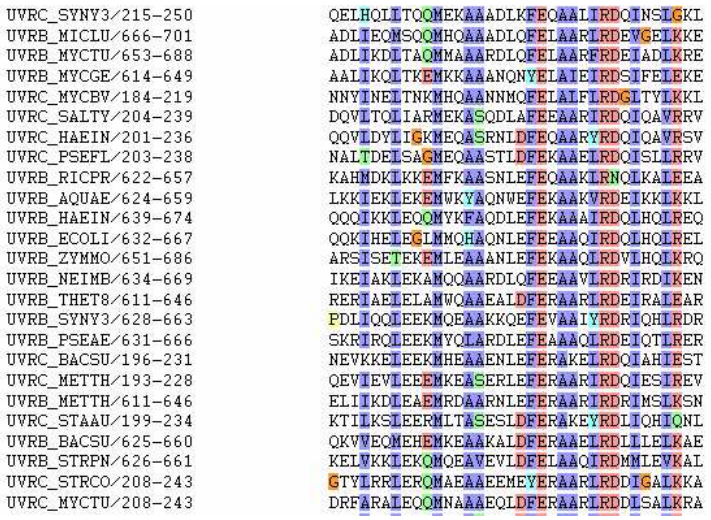


Figure 6: The Pfam multiple sequence alignment output for UV radiation repair proteins uvrB and uvrC. Since these proteins are present in many organisms, the alignment is shown for the first 80 amino acids with a high number of sequences which are representative of species that have successfully had their uvrB/uvrC proteins sequenced.

research which is elaborated on in Chapter 5.

2.2 Graph Theory

In 1736 Leonhard Euler, one of the most famous mathematicians of all time, solved a problem known as the Königsberg Bridge problem [18]. The problem consisted of finding a path to walk over the seven bridges of the East Prussian city once called Königsberg, with the requirement that no bridge be crossed more than once (in either direction). A diagram of the problem is given in Figure 7.

In doing so, he opened a field of mathematics called Graph Theory that has since been rapidly developed and become critical to our analysis of the world around us. In this section, we explain the basics of graphs and discuss a special type of graph which will be important to our structural bioinformatics approach to DNA repair protein conformation prediction.

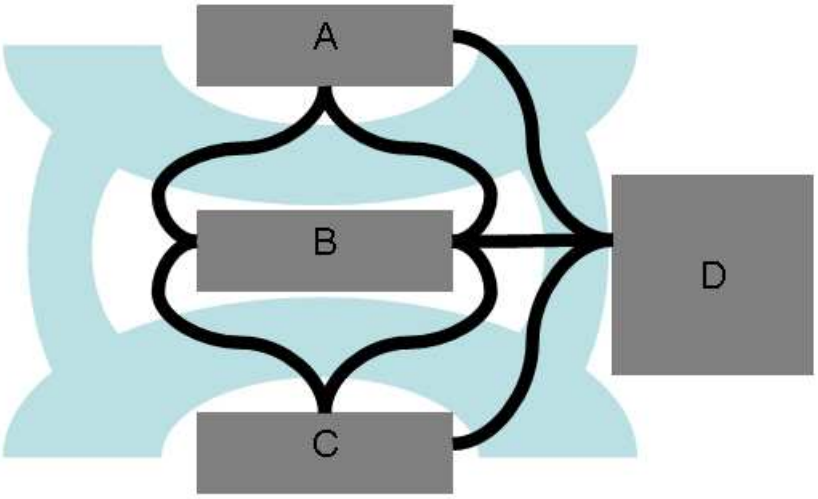


Figure 7: The Königsberg Bridge problem. There are four populated areas labeled A-D (gray) separated by rivers (blue), and the problem is to cross each of the seven connecting bridges (black) exactly once, finishing the journey at the same place it was started.

2.2.1 Fundamental Graph Theory and Notation

When Euler solved the Königsberg Bridge problem, he did it by introducing a *graph*, which is a collection of points and links connecting the points. More formally, we say that a graph $G = (V, E)$, where V is the set of points, more formally called *vertices*, and E is the set of connecting links between the points, more formally called *edges* and defined as $E = \{v_1v_2 | v_1, v_2 \in V\}$.

Graphs arise in countless applications from goods transportation, plumbing design, and national electric grids to metabolic diagrams, image processing, and security systems. By transforming a problem into a series of points and connections between points, we can infer properties about the system overall. For example, we can examine the connectedness of the power grid in a nuclear reactor which is supplying electricity to a city in order to find out how many points connecting the reactor to the city can fail or be lost before the city loses power. As a more biological problem, we can construct a graph of metabolites and their reactants, and from the graph determine which compounds are critical for a particular species, or which compounds lead to the evolution of new

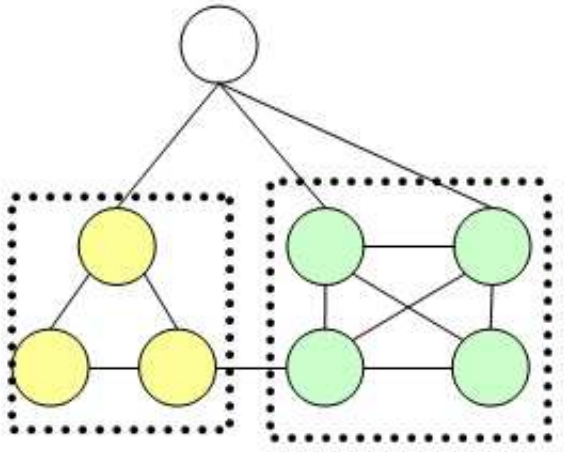


Figure 8: A graph that illustrates cliques. One clique colored in yellow has three vertices, and another clique colored in green contains four vertices. Notice that this graph is not a complete graph because there are vertices that do not connect to each other, such as certain vertices between the two colored cliques.

compounds.

Graph theory and properties of graphs are now firmly established in literature. For example, graphs can have edge direction as in the goods transportation example, or graphs can have edge weights similar to the capacities of pipes used in plumbing. A simple treatment of Graph Theory is available in [36], with a more advanced discussion in [18]. An excellent application of graph theory to real-world problems in social, scientific, economic, and technological disciplines, to name only a few, can be found in the general audience book by Barabási [8].

2.2.2 Cliques

There is a special type of graph called a clique which will be a critical tool for the work presented in Chapter 3. A clique is a type of graph where all of the vertices are connected to each other, much like a complete graph. Where cliques and complete graphs differ is that cliques can exist as subgraphs of a larger graph and need not satisfy the property for all vertices in an original graph. Formally, for a graph $G = (V, E)$, then $V' \subseteq V$ is a clique if for all $v_1, v_2 \in V'$ we have $v_1 v_2 \in E$ $(v_1 \neq v_2)$.

A graph G is given in Figure 8 with two cliques inside of it. The real-world

phenomena that cliques convey is the sense of interconnectedness amongst all things in a clique. Going back to our power grid example, a clique in the graph representing a power grid means that any given power distribution line connects to any other distribution line, and no matter what node (vertex) in the grid we take down or experience failure with, the system will continue to be able to deliver power through its remaining connections.

In this thesis, we will use cliques to replicate the sense of interconnectivity among atoms in a protein. Though the details will be given more extensively in Chapter 3, we mention here that we will use an edge-weighted graph and extract its maximum clique, or in other words, that we will find all of the cliques that are subgraphs of a connectivity graph and extract the largest one because it represents maximum connectivity in the protein under consideration.

2.3 Machine Learning

If you were to take a survey of how many people can write the cursive alphabet characters Q or Z at any given moment, you will likely find that a number of people struggle in this task. Yet if you ask them to take a collection of human drawn guesses combined with computer printouts of the cursive letters Q, Z, and other random cursive letters, we can expect that they can distinguish the correct characters from the incorrect characters fairly well. Why is it that they cannot necessarily draw the characters themselves yet they can recognize them?

Aside from the cognitive reasons dealing with expression and creation in the human brain, the reason that they can recognize the characters is that they have learned from a set of examples which were provided along with the correct answer for each example. It could be that the examples were given to them in elementary school, or two minutes before the experiment. The key is that a set of examples have been provided for them to *learn* from, and after learning on a number of examples, they can more accurately classify an unknown or new object. In our experiment of identifying cursive Qs and Zs, the previous experiences, whether they are human drawings or computer printouts, are the learning data, and another set of drawings and printouts will be our new data (test data) to test out the ability to identify Qs and Zs accurately. Figure 9

Figure 9: A computer font that illustrating the similarities between Q, Z, S, 2, 5, and $. Curvature alone is an inadequate feature to automatically distinguish and identify these six characters, since they all share similar curvatures. In learning, many features are used to successfully classify objects.

illustrates this point by showing a number of characters similar to and including Q and Z.

It is this same process that Johannes Kepler performed in his study of planetary motion. After performing a series of transformations on the positional patterns of the planets [33], in the beginning of the 17th century he uncovered what are now known as Kepler's Laws of Planetary Motion. Similar to how we identify new Qs and Zs using a collection of previously available drawings and printouts, Kepler identified new patterns in the numbers after using a collection of data recorded decades earlier [33].

This process of identifying new objects from training sets of existing objects is more generally known as *pattern recognition*, and when the concept is applied using a computer program to find patterns and make guesses from them, it is called *machine learning*. Since computers can be trained to recognize patterns and then make a decision about a new piece of data, they can act similar to human beings or other biological species.

As computers have drastically increased in speed over the past decade, their

application to real-life situations has also increased. In Chapter 4, we will use this machine learning approach to identify patterns in DNA repair proteins, so that we can identify novel DNA repair proteins in newly sequenced genomes or unannotated existing genomes. By doing that, we can increase our range of knowledge about the types of DNA repair, look for existing or new patterns in additional data, and work on a network construction that will be discussed in Chapter 5.

In order to be more concrete about machine learning, in the remainder of this chapter we discuss the concept of linearly separable data and we consider a particular implementation of machine learning known as the Support Vector Machine.

2.3.1 Linearly Separable Data

The Q-Z example at the beginning of this section is a fairly simple instance of a more general problem: distinguishing between two classes of objects. If we are looking at a particular property of the objects, then we may describe the two classes of objects as the classes of *have*s and *have not*s, since objects of one class will have the property, and objects of the other class will not. In our example of distinguishing Qs and Zs, we might have said that objects that *have* two loops and a tail are cursive Qs, and objects that do not have two loops and a tail are not Qs. After examining the characters in Figure 9, we might have instead suggested the double loop criteria for Z rather than for Q.

Generalizing this situation more, we can say that we want to distinguish between two classes which we will call the *positive* and *negative* datasets. In most situations, we are concerned with identifying new positive examples, such as new cases of cancer or new buildings with factors that could cause it to collapse in an earthquake.

Human beings try to come up with some "clear-cut" criteria for a decision problem. In a sense, they "draw a line" so that when the line is crossed, a definite decision is made. A computer can analogously do the same thing, but since a computer has no intuition, there must be a precise way to specify how it "draws the line." This is the concept of linearly separable data.

If we look at the red dots and blue dots on the left side of Figure 10, we

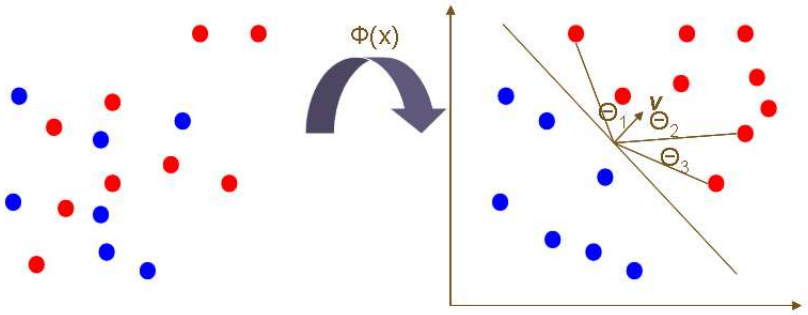


Figure 10: A simple problem of trying to separate red dots from blue dots. On the left side of the figure, the dots are mixed about such that it is difficult to mathematically describe a way to separate them. After a transformation $\phi(x)$, the data is transformed to its representation on the right side of the figure, and has become easily separable.

see that it is quite difficult to create a line to distinguish the two data sets. A transformation to separate the dots so that they can be linearly separated similar to the right side of Figure 10 will be discussed in Section 2.3.2.

Mathematically, if two objects v_1 and v_2 represented as vectors are similar, then their dot product $< v_1, v_2 >$ should be a positive value because of the fact from linear algebra that $\cos\theta = \frac{v_1 v_2}{|v_1||v_2|}$, where $\cos\theta$ is the cosine of the angle θ between the two vectors. Referring to the right side of Figure 10, this means that we want to find the vector v so that the red dots (which we will treat as positive examples) make angles with v that are $\leq 90°$, since high school trigonometry teaches us that $\cos\theta \geq 0$ for $-90° \leq \theta \leq 90°$. The angles θ_1, θ_2, and θ_3 are three such angles made by the angle between v and the red dots.

For the moment what is important is that if there is a way to transform our data (which is a series of characteristics of an object) such that we can separate the data into two classes, then we can predict which of the two classes an unknown object belongs to.

This problem of finding a line to separate linearly separable data was first solved by a method called the Perceptron Algorithm, developed by Rosenblatt in 1956 [15]. The key element of the Perceptron Algorithm is the separating line $ax + by = c$ (called a hyperplane in three dimensions or above), derived

from the input vectors (training examples). Non-separable data requires a more advanced treatment, which is the subject of the next section.

2.3.2 Support Vector Machines and Kernel Methods

After Rosenblatt's Perception algorithm was successfully demonstrated on linearly separable datasets, the natural question was raised about what to do with datasets that were not linearly separable. A fairly intuitive solution is to transform the data by adding extra dimensionality in such a way that the data becomes separable. However, as data increases in dimensionality (the number of characteristics we are measuring on an object), it becomes difficult, in terms of computational cost, to compute high-dimensional hyperplanes that can separate the data into two classes. Furthermore, applied to real world problems such as handwriting or facial recognition where there may be tens of thousands of images in the dataset, high-dimensional computations are simply too costly.

In the final decade of the 20th century the machine learning methodology started to make use of the kernel method, which was developed in mathematical analysis and is used in mathematics and physics for dealing with convolution or other topics [26]. While the details of kernel methods in machine learning are well established both conceptually and mathematically in [15, 33], for the purpose of understanding how this helps in DNA repair, here we only mention the core idea. In short, a kernel method computes the product of two vectors as though they were in fact transformed into a higher dimensional space, but does so by using some mathematical function $\phi(\mathbf{x})$ satisfying $K(\mathbf{x}, \mathbf{y}) =< \phi(\mathbf{x}), \phi(\mathbf{y}) >$, where $\mathbf{x}$ and $\mathbf{y}$ are vectors ($\mathbf{x} =< x_1, x_2, \dots >$), and $< \mathbf{x}, \mathbf{y} >$ denotes their inner product (also called the dot product). This is the meaning of $\phi(\mathbf{x})$ in Figure 10.

A trivial example of a kernel method is the linear kernel, defined using $\phi(\mathbf{x}) = \mathbf{x}$. Here, $K(\mathbf{x}, \mathbf{y}) =< \phi(\mathbf{x}), \phi(\mathbf{y}) >=< \mathbf{x}, \mathbf{y} >$. Another example is the polynomial kernel, $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d$, for arbitrary values of c and d, where $\mathbf{x} \cdot \mathbf{y}$ again means the dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$. For example, for $c = 0$ and $d = 3$,

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^3 = (x_1 y_1 + x_2 y_2)^3$$

$$= (x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2)(x_1y_1 + x_2y_2)$$

$$= x_1^3y_1^3 + x_1^2x_2y_1^2y_2 + x_1x_2^2y_1y_2^2 + x_2^3y_2^3 + 2x_1^2x_2y_1^2y_2 + 2x_1x_2^2y_1y_2^2$$

$$= (x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, \sqrt{2}x_1^2x_2, \sqrt{2}x_1x_2^2) \cdot (y_1^3, y_1^2y_2, y_1y_2^2, y_2^3, \sqrt{2}y_1^2y_2, \sqrt{2}y_1y_2^2)$$

$$=< \phi(\mathbf{x}), \phi(\mathbf{y}) >$$

for $\phi(\mathbf{x}) = (x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, \sqrt{2}x_1^2x_2, \sqrt{2}x_1x_2^2)$. This example was provided for two elements (characteristics) per object, but we could extend it to any amount. Similarly to linear data as explained in Section 2.3.1, when two objects (vectors) are similar, the angle between the two vectors is low, and the value of the kernel computation is high. The higher the value, the higher the similarity, and the more confidence we have in distinguishing like objects.

How does any of this help us? By finding a kernel, we can skip the transformation of data to high-dimensional space and use only the kernel to do the computations. This saves us (or in most cases, the computer) an enormous amount of time when data becomes complex. In the case of DNA repair proteins, we will sometimes use hundreds or thousands of features to compute their similarity. By using kernels such as the linear and polynomial kernels, along with several others to be introduced in Chapter 4, we can evaluate thousands of proteins very quickly.

The final piece we need to do the research in Chapter 4 is an application which can use these kernel methods. The Support Vector Machine (SVM), which has become extremely popular since the start of the 21st century, is a technique which is mathematically and statistically well founded [15], and is superlative to the Perceptron for many reasons. The first reason is that it works well on non-linearly separable data, such as the data in Figures 9 and 10. The second reason is that the standard measure of similarity, the vector dot product $< \mathbf{x}, \mathbf{y} >$, can be substituted by any kernel $K(\mathbf{x}, \mathbf{y})$ which is more appropriate for the problem at hand. Finally, the SVM method can adapt to and "disregard" data (apply a soft margin to data) that simply is too difficult to include when considering how to build the separating hyperplane.

The SVM technique has been conistently shown good results, for example in bioinformatics [11]. As we will show in Chapter 4, the SVM method and

kernel methods are quite capable of identifying what constitutes a DNA repair protein and what does not.

Chapter 3 Structural Bioinformatics approach - Side Chain Packing

3.1 Introduction

The protein side chain packing problem (SCPP) is an important problem in bioinformatics. The problem consists of, given a protein's backbone atoms (whether predetermined via throughput techniques or computational prediction), predicting the location of the resulting side chain atoms, concluding in the prediction of a protein's final conformation. The problem can be visualized by considering the two proteins shown in Figure 11. Since a protein's structure affects its function, the ability to correctly predict a protein's structure affects the ability to correctly predict its corresponding function. Correct prediction of protein function paves the way to designing new chemicals and drugs in order to cure disease and illness.

The SCPP problem has been proven by Akutsu to be NP-hard [1], so unless P = NP, methods to find the optimal solution to this problem in polynomial time are unlikely to exist. Hence, the focus of this chapter and attention of researchers has been in how to develop other methods to find the optimal solution to the problem. These alternative schemes can be broadly classified into the two categories of heuristic and deterministic methods. For example, rapid tree decomposition [38] and integer programming approaches [19] are two deterministic methods, while SCAP [37] is one heuristic method to try and solve the problem.

In this chapter, we examine the SCPP from the heuristic aspect of breaking it into subproblems, with each problem, when solved, contributing to the overall solution of the original prediction problem. The approach, first given by Bahadur *et. al* in 2005 [5] for linear subpolymers of even length, is generalized in this chapter. Also presented in this work are several new methods for protein subdivision which are more realistic in considering a protein's spatiality.

Many of the methods developed for SCPP transform the problem into a graph [5, 38]. How the graphs are utilized is where the methods and results differ. As in the method of [5], the methods in this chapter also transform the

problem into a graph in order to find a maximum weight clique, given an edge weighting function that reflects the frequency of contact pairs in a database of proteins. We include a characterization of each graph method so that methods can be compared. Compared with other complicated methods, the algorithms presented here are simple enough that they can be readily implemented for evaluation or usage.

The rest of this chapter is organized as follows. The problem is concretely defined with an overall solution given in Section 3.2, along with a discussion of several methodologies for graph generation in Section 3.3. The results of those methodologies are presented and discussed in Section 3.4, including a revision of the results in [5]. In Section 3.5, we conclude with additional discussion of the results, including a fair assessment of the strengths and weaknesses of the approaches presented.

So as to not confuse the reader, we note that we interchangeably use the terms protein and polymer, with a subpolymer being a portion of the protein input. We also note that the main objective of this chapter is not to correct [5], but more importantly to consider spatially realistic methods of protein subdivision for side chain packing.

3.2 Problem Definition

Computational protein structure prediction methods often are done in two phases. The first phase is threading, or predicting the location of only the backbone atoms. The second phase is side chain packing, or predicting the location of a protein's side chain atoms based on its backbone atoms. Accurate chemical research is dependent on successful execution of both of these steps. Here, let us describe the overall framework for understanding the side chain packing problem, along with a general description of an algorithm for solving it.

3.2.1 Problem

Concretely, the SCPP can be phrased as this: given a protein of N amino acids and the atom positions for the backbone atoms in those acids, predict the physical location of each atom present in the side chains of the amino acids. Let

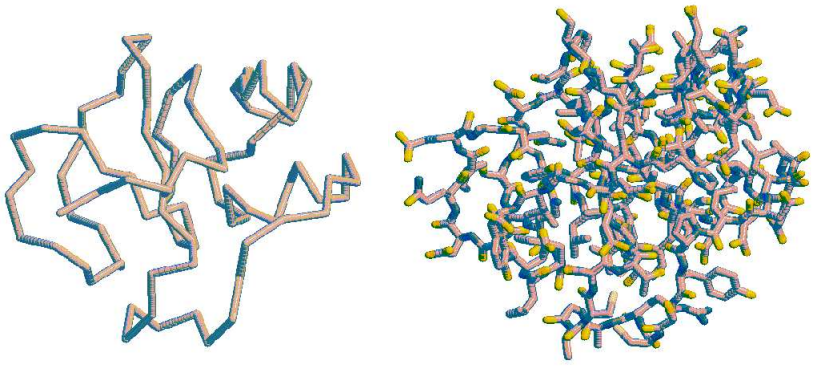


Figure 11: The side chain packing problem. The left image represents an input protein whose backbone structure (collection of atom positions) is known. The objective is to use the given backbone atom information and computationally predict the final positions of the side chain atoms that would normally appear in a real protein. The image on the right is the predicted final conformation of the input protein on the left.

us number the residues (amino acids) of the protein $a_1, a_2, a_3, \cdots, a_N$. Each of these acids can be rotated into a particular conformation (rotamer) about the χ_1 axis, making a torsion angle between the C_α atom and the side chain. In [13, 38], rotamer libraries are used to define the rotation angles through which particular acids may rotate, though similar to [5], they are not used here. By excluding rotamer libraries, algorithm implementations need not be concerned with details of rotamer libraries, and additionally, the algorithm is applicable to proteins which are not covered by rotamer libraries. As will be shown, fears that an excessive number of rotamers are generated are quelled by eliminating many rotamers before graph generation.

In this chapter and in [5], we wish to divide the protein P into p subpolymers $P_1, P_2, \cdots, P_p$, where obviously $p < N$. For each subpolymer P_i, rotamers are generated for each amino acid $a_j \in P_i$, beginning at a random angle θ_j for each acid a_j, and generating a new rotamer at every $\frac{2\pi}{R}$ degrees, where R is the specified number of rotamers to generate per amino acid. Hence we generate rotamers $r_{u,j,v}$ for subpolymer u's jth residue rotated $\theta_j + (2\pi v/R)$ degrees.

Note again that initial rotation angles θ_j are independently generated for each residue $a_1, \cdots, a_N \in P$. Following the method of [4], the value of R is set to 20, but could be adjusted as needed for more or less precision.

Here, it becomes easy to understand why this combinatorial optimization problem is intractable. For a protein of N amino acids, we generate R rotamers, initially resulting in R^N configurations and, as to be explained, NR graph vertices to search. As the size of the protein increases to considerable size (e.g., the PDB Data Bank [10] structure *1xwl* has 580 residues), our search space grows rapidly, especially if we increase the polymer size and the number of rotations R simultaneously. Using the same example, this generation process immediately creates an initial search space of more than $3 * 10^{754}$ possible configurations, and 11,600 graph vertices for the protein *1xwl*.

We transform the problem into a graph theoretic-problem. This is done because there is a sense of spatial interconnectedness amongst the atoms in a protein, and we need our solution to reflect this. For each of the rotameric configurations we have generated, we let each individual configuration (of all atoms in an arbitary rotamer) be represented by a vertex in a graph, as shown in Figure 12. To capture the interconnectedness of a real protein, we assign weights to the edges between vertices in our graph using a weighting function. The interested reader is directed to [31] for the details of the weighting function, but in short, the function uses a database of known protein structures and calculates an all-atom distance dependent probability function for assigning weights. Returning to our problem, we wish to find the maximum edge-weight clique in the graph (a clique is a connected graph, c.f. [16] and Figure 8 in Section 2.2.2) which represents the most likely final configuration of the protein being calculated. In order to ensure that the resulting clique is a spatially realistic polymer, we must check for consistency, as detailed below.

3.2.2 Consistency

After all rotamers are created in a sampling step, three checks are necessary to ensure that the rotamers and their respective atoms present in the resulting graphs are those that would not physically collide.

As a preprocessing step, we remove those rotamers generated for the acids

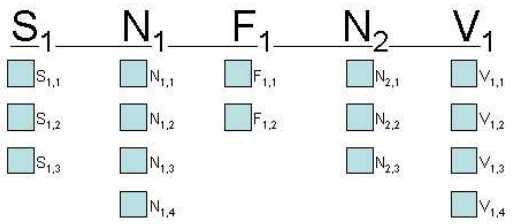


Figure 12: The side chain packing problem recast as a graph-theoretic problem. For each acid A_i, we generate up to R rotamers $A_{i,1}, \ldots, A_{i,R}$. In this example $R = 4$, and those acids in with less than 4 rotamers have had some rotamers eliminated through consistency checks, as explained in Section 3.2.2.

glycine, alanine, and proline, following the method of [5]. These three acids typically do not have rotamers, and experiments show that this decreases the total number of rotamers to search by 15-25%.

The first check that is done is to eliminate those rotamers which collide with the fixed backbone atoms. Let $r_{u,j,v}$ be a rotamer in our search space. We check each side chain atom of $r_{u,j,v}$ against the backbone atoms of the entire input polymer P, and if any of the atoms collide, then we remove the rotamer from our list of candidate vertices. A rotamer is defined to collide with the backbone if the distance between any side chain atom of the rotamer and any atom of the backbone is less than 1Å.

The second check that occurs is to verify the consistency of the rotamers in the subpolymer itself. In other words, we eliminate those rotamers which collide with other rotamers in the same subpolymer. We call this "internal consistency," and in symbols we mean that if a rotamer $r_{u,j,v}$ collides with a rotamer $r_{u,k,x}$ for any k and x (except for $r_{u,j,v}$ itself), then the rotamers $r_{u,j,v}$ and $r_{u,k,x}$ are internally inconsistent. In a graph theory sense, this means setting the edge weight between the two edges to zero ("removing the edge") so that the two vertices (rotamers) in question simultaneously cannot occur in the maximum clique for the subpolymer. Two rotamers are defined to collide if any one of the atoms from one rotamer is within 4Å of the atoms from another rotamer.

The final check that must occur is that those rotamers which are internally

consistent must also be externally consistent, or that they must not collide with rotamers from other subpolymers. Our distance constraint remains at 4Å. Let $r_{u,j,v}$ be a rotamer that collides with a rotamer $r_{w,k,x}$ from another subpolymer. Then because $r_{u,j,v}$ creates a collision with a rotamer in another subpolymer, it cannot possibly be in the final solution, and hence, we remove all edges connected to rotamer $r_{u,j,v}$. Since we are considering only the subpolymer to which $r_{u,j,v}$ belongs to, we do not remove the edges connected to $r_{w,k,x}$, as those edges will be removed when we consider the subpolymer to which $r_{w,k,x}$ belongs to.

If a rotamer successfully passes these three checks, then it is a physically feasible rotamer. Since we are breaking the original polymer P into subpolymers P_i, we only weight the edges for which two rotamers are in the same subpolymer P_i (i.e., $r_{i,j,v}$ and $r_{i,k,x}$). Readers should see [31] for the details of the edge-weighting function.

3.2.3 Algorithm

As each subpolymer graph is made, the remaining step in the per-subpolymer algorithm is to find its maximum edge-weighted clique, and finally to use the union of subpolymer cliques as the solution to the original SCPP. We use the efficient WCQ weighted-clique algorithm developed by Suzuki et. al [34], which in turn makes use of the MCQ maximum clique algorithm developed by Tomita and Seki [35]. Interested readers can find the details in [34, 35], but we mention here that in our experiments a graph represented by a symmetric adjacency matrix of 3365×3365 entries (rotamers) had a maximum clique found in 1 second, with a maximum edge-weight clique found in 2 seconds (using a 3.2 GHz Pentium 4 Processor with 1.0 GB RAM).

Our overall algorithm to solve the SCPP is shown in Figure 13. We present a number of its details in order to accurately give a time complexity analysis on certain steps, leaving out constant time actions (that are typically program internal administrative duties).

Our analysis is as follows. Steps 1,3, and 8 can be completed in $O(N)$ time, where N is the number of amino acids in the protein. Steps 2,4, and 5 make use of at most R rotamers per acid, and can be done in $O(NR)$ time. Step 6,

1. Load PDB.
2. Create all rotamers.
3. Trim redundant rotamers.
4. Trim backbone colliding rotamers.
5. Assign ID number to all remaining rotamers.
6. Create subpolymers and respective graphs.
7. Find cliques for subpolymer graphs.
8. Combine subpolymer clique solutions and output final solution (protein).

Figure 13: Algorithm for computing side chain packing of a protein.

the crux of the chapter, depends on the subpolymer creation method, and is analyzed in the Methods section below. Finding a maximum clique is known to be NP-hard [16].

3.3 Methods

In this section, we show three different methods for breaking a protein into subpolymers, and characterize the resulting graph sizes for each of their respective graph generation methods. Recall from Section 3.2.1 and Step 2 of Figure 13 that the initialization process automatically generates NR vertices, with some portion of them removed for redundancy in Step 3.

3.3.1 Linear Polymer

The most intuitive subpolymer is a linear subpolymer, shown in Figures 14 and 15. It is created by simply breaking up the amino acid sequence at specified residues and assigning the resulting fragments to subpolymers. As a toy example, if we have a protein with an amino acid sequence $P = SNFNVCRLPGTP$, and we break it into 3 linear subpolymers, one possible assignment could give the resulting subpolymers $P_1 = SNFN$, $P_2 = VCRLPG$, $P_3 = TP$. The key to linear subpolymers is that their sequence order is preserved. Analysis of the graph size is then as follows: in the first subpolymer P_1 containing s_1 residues and hence a maximum of s_1R rotamers, it must check for consistency against itself and all remaining ($\leq NR$) rotamers in the original polymer P. As the external consistency check between subpolymers P_1 and P_2 has been done in this pass, there is no need for P_2 to check its external consistency again with P_1,

and hence, the second subpolymer only needs to check subpolymers P_2 through P_n, processing a total $s_2R * (N - s_1)R$ rotamers, where s_2 denotes the number of residues in subpolymer P_2. Continuing in this fashion, the number of vertices processed over all subpolymers is

$$\leq \Big(s_1R(NR)\Big) + \Big(s_2R(N - s_1)R\Big) + \Big(s_3R(N - s_1 - s_2)R\Big) + \cdots + \Big(s_pR(s_pR)\Big)$$

$$= \Big(\sum_{i=1}^{p}[s_iR][(N - \sum_{j=1}^{i-1} s_j)R]\Big).$$

In this framework, if we consider the formulation in [5], the number of residues processed over all subpolymers of even size s_0 is

$$= \Big(\sum_{i=1}^{p}[s_0R][(N - \sum_{j=1}^{i-1} s_0)R]\Big)$$

$$= \Big(s_0R^2\Big[\sum_{i=1}^{p} N - \sum_{i=1}^{p}\sum_{j=1}^{i-1} s_0\Big]\Big)$$

$$= \Big(s_0R^2\Big[Np - \frac{s_0p^2}{2} + \frac{s_0p}{2}\Big]\Big),$$

and since all polymers are of size $s_0 = \frac{N}{p}$, the total number of graph vertices generated over p even length subpolymers is

$$\leq \Big(\frac{1}{2}N^2R^2\Big[\frac{p+1}{p}\Big]\Big).$$

We can expect the non-uniform length subpolymer graph generation method will create a similar number of vertices, using the fact that $N = s_0p = (s_1 + s_2 + \cdots + s_p)$.

As the number of subpolymers increases, the total amount of graph vertices generated is reduced, until we reach the limit

$$= \lim_{p\to\infty} \frac{1}{2}N^2R^2\Big[\frac{p+1}{p}\Big] = \frac{1}{2}N^2R^2.$$

This clearly establishes the advantage of the linear subpolymers proposed here and in [5], which have a total number of vertices kN^2R^2, where $\frac{1}{2} \leq k \leq 1$, over the technique in [6], which is always N^2R^2.

A drawback of the algorithm in [5] is that by dividing a polymer into even

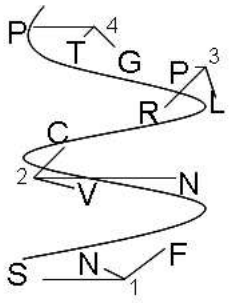


Figure 14: In this example, the sample protein $P = SNFNVCRLPGTP$, which resembles a beta sheet, is divided into four even-length subpolymers, as in [5]. The numbers indicate resulting subpolymers.

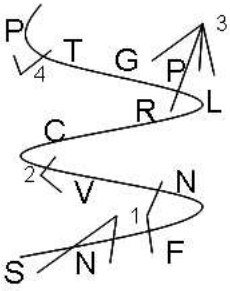


Figure 15: The same protein as in Figure 14, but broken into biologically significant clusters (subpolymers) where turns occur. Subpolymers are no longer of the same length.

length divisions, we fail to solve the problem in a biological sense. Instead, in order to improve the technique's applicability, one can divide a protein into linear subpolymers at biologically significant residues using data from the UniProt [7], Pfam [9], and Interpro [29] databases. Results of computing side chain packing using linear subpolymers and this data are given in Section 3.4.

3.3.2 Centroid Subpolymers

Since a protein typically does not stay linear and flat, alternative subpolymer methods are also useful for solving the SCPP by decomposition. Our next approach to division by subpolymer division was done using the centroid of a protein, illustrated in Figure 16. We find the centroid of a protein by adding the respective X, Y, and Z coordinates of every atom in the protein and dividing by the total number of atoms. We then determine how many residues are located on each side of the axes running through the centroid, and use the axis with the minimum residue difference as the axis to break the original polymer into two subpolymers $P_{\leq}$ and $P_{>}$. Note here that the residues no longer need to preserve order as they did in a linear subpolymer.

Here, the analysis is fairly simple. We have two subpolymers, of size s_1 and s_2 such that $s_1 + s_2 = N$, and we must check each polymer for internal consistency and external consistency against the other subpolymer. It is obvious that this

requires $\leq (s_1^2R^2+s_1s_2R^2)+(s_2^2R^2+s_2s_1R^2)$ rotamer consistency checks for both graph generations, which is equivalent to the number of vertices as the N^2R^2 method in [6], because the centroid method is equivalent to twice generating the first subpolymer of a linear subpolymer. A perfect centroid evenly divides the amino acids into two even-length subpolymers, though experiments rarely yielded this. Where this algorithm becomes useful is in its consideration of the spatial arrangement of atoms. Results from this type of graph generation are given in Section 3.4.

3.3.3 Distance Bin Subpolymers

Our final method of creating subpolymer graphs is by using distance bins. In this approach, we find the maximum and minimum X, Y, and Z coordinates among all atoms in the protein. We then take the axis with the maximum width (i.e., maximum of $x_{max} - x_{min}$, $y_{max} - y_{min}$, and $z_{max} - z_{min}$) and divide it amongst b bins of even length. We put each amino acid into the corresponding distance bin (thus we do not preserve order), using the centroid of that acid with respect to the axis being used for binning. The process is shown in Figure 17.

For subpolymer graph generation, we need only consider internal consistency and the two adjacent subpolymers for external consistency, rather than the entire original polymer. In practice, we add fictitious empty subpolymers to both ends of the protein so that the subpolymer graph generation works like a sliding window on bins $1 \ldots b$. For each call to the graph generation method on a subpolymer of size s_i residues with neighbors of size s_{i-1} and s_{i+1}, we need $\leq (s_iR(s_{i-1}R + s_iR + s_{i+1}R))$ consistency checks. Taken over b bins, we have (where s_0 represents our fictitious empty subpolymers)

$$\leq s_1R(s_0R+s_1R+s_2R)+s_2R(s_1R+s_2R+s_3R)+\cdots+s_bR(s_{b-1}R+s_bR+s_0R)$$

$$= R^2(s_1^2 + 2s_1s_2 + s_2^2 + 2s_2s_3 + \cdots + 2s_{b-1}s_b + s_b^2)$$

$$= R^2(N^2 - 2\sum_{i=1}^{b-2}\sum_{j=i+2}^{b} s_is_j)$$

total graph vertices to check over all subpolymers that are created. It is obvious that the distance bin method can create no more vertices than the linear

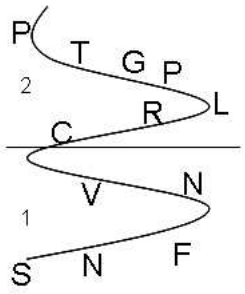


Figure 16: The example polymer is divided into two subpolymers at its centroid. In this two-dimensional example, the Y axis is used for doing the division work. Though the order of residues and their respective rotamers is preserved in this example, it would not be preserved if the X axis were used.

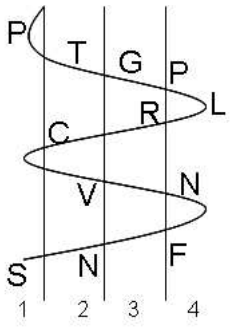


Figure 17: The example polymer $P = SNFNVCRLPGTP$ is broken into four distance bins ($P_1 = SP$, $P_2 = NVCT$, $P_3 = RG$, and $P_4 = FNLP$) along the X axis. The subpolymers have differing size and no longer preserve the original protein's sequence order.

supbolymer or centroid method (unless $b = \{1, 2\}$). A value of $b = 1$ or $b = 2$ agrees with the N^2R^2 result of the linear and centroid methods, as it intuitively should.

Interestingly, when we observed the numbers of acids that fell into each distance bin, we generally found the values to follow a Gaussian distribution, though we make no further analysis based on this observation.

3.3.4 Completed Analysis

Combing the analysis from Section 3.2 with the methods in Section 3.3, it can be seen that the graph generation step is the most most demanding, except for the distance binning method (as long as $b \neq \{1, 2\}$). Even in the case of processing a whole protein as a single subpolymer (for linear and distance bin cases), the worst case number of rotamer comparisons is N^2R^2.

3.4 Experimental Results

Side chain packing accuracy can be predicted by using root mean squared deviation (RMSD) scores. RMSD is calculated by measuring the difference between

a reference object (in our case, a known protein conformation) and a test object (in our case, the protein's predicted final conformation), accommodating for any differences in signs that may occur. For our side chain packing experiments, we define RMSD to be

$$RMSD = \sqrt{\frac{\sum_{atoms}\left(\sqrt{(x_r - x_t)^2 + (y_r - y_t)^2 + (z_r - z_t)^2}\right)^2}{n_a}} \quad ,$$

where x_r is the x-coordinate of an atom in the reference protein, x_t is the x-coordinate of the atom corresponding to x_r in the test protein, and n_a is the total number of atoms in the protein. The meanings of y_r, y_t, z_r, and z_t are analogous, and $\sum_{atoms}$ indicates to sum over all of the atoms in a protein. In the formulation of RMSD above, it is visible that there is a redundant calculation when taking the square of the inner square root. However, this is the calculation we employ because it follows the traditional format for calculating standard deviation when the mean is replaced by the known protein. Hence, the inner square root is applied to atom distance, and the outer square root is applied for variance.

After Bahadur *et. al* reported in [5] RMSD scores for an even-length protein division method, a mistake in the code was discovered that affected the final results of the method. Fixing this code mistake, several polymers were re-evaluated, and the result is that several polymers no longer achieve maximum cliques. We display the results of the experiment using corrected code in Table 2, where "dnf" denotes that a maximum clique was not achievable using the specified number of even-length subpolymer divisions. This new result is not a setback but rather a confirmation that polymers cannot simply be broken apart at random locations without some biological consequence. Furthermore, this motivated the study presented here for breaking polymers into pieces based on biologically relevant locations or more realistic spatial considerations.

In order to evaluate the methods proposed in this chapter, we used the same set of proteins as in [4, 5], and set a time limit of 5 minutes for finding a maximum clique. The results of the side chain prediction are shown in Table 3. Overall, at least one of the methods succeeds in finding a solution to the SCPP

Table 2: Side chain packing results reported in [5] (in parentheses), and adjusted values after program code corrections. "dnf" means the protein did not finish computing in under 30 minutes, a limit set based on the packing runtimes in [5]. The column headings indicate the number of even-length subpolymers p the original protein was divided into. All calculations are RMSD in Å.

PDB	Residues	p = 2	p = 3	p= 4	p = 5
1xwl	580	(1.66)dnf	(1.56)dnf	(1.60)dnf	(1.74)dnf
1gof	639	(1.63)dnf	(1.46)dnf	(1.69)dnf	(1.70)dnf
1biy	689	(1.53)dnf	(1.64)dnf	(1.77)1.84	(1.72)1.87
1aa6	696	(1.58)1.87	(1.68)1.80	(1.61)1.83	(1.70)1.82
1a81	812	(1.56)dnf	(1.63)dnf	(1.63)dnf	(1.75)dnf
1lnh	836	(1.50)1.93	(1.63)dnf	(1.64)1.87	(1.55)dnf
1fiy	874	(1.42)dnf	(1.53)dnf	(1.63)dnf	(1.54)dnf

for a protein, with many proteins being successfully calculated for multiple methods. A number of proteins had improved accuracy as the number of bins increased, signifying that a particular subpolymer could realistically consider only its immediately adjacent neighbors to predict the side chain conformations.

To put this study into perspective on a realistic set of proteins for an application, we tested a small subset of proteins that were extracted by querying the Protein Data Bank [10] with the phrase "DNA excision repair." We imposed the constraint that we compare our result against only the first model of a PDB, though we include all chains. The results of the proposed methods on the protein set are shown in Table 4. Note that many DNA excision repair proteins are smaller than the test sets used in [4, 5], and subsequently, several proteins could be calculated using all three techniques.

It is to note here that while many proteins did not finish within the 5 minute time limit, if we let the clique finder execute without a time restriction, it would eventually find and return a clique corresponding to the solution for a particular subpolymer. For the purposes of speed, we set our limit low.

Table 3: The results, in RMSD Å, of computing the same proteins used in [4, 5], applying the methods described in this chapter, where b is the number of distance bins used. "dnf" signals that a maximum weight clique was not found within a time limit of 5 minutes. "DBAve." is the average result for the distance bin method using 5-20 bins.

PDB	Residues	Linear	Centroid	b = 10	b = 11	b = 12	b = 13	b = 14	b = 15
1xwl	580	2.036	2.055	1.897	1.887	1.804	1.818	1.826	1.873
1gof	639	1.876	dnf	dnf	dnf	dnf	1.742	dnf	dnf
1biy	689	dnf	dnf	1.850	dnf	dnf	1.738	1.721	1.806
1aa6	696	dnf	dnf	dnf	dnf	dnf	dnf	dnf	1.733
1a81	812	dnf	dnf	dnf	dnf	dnf	dnf	dnf	1.823
1lnh	836	dnf	dnf	dnf	1.806	1.855	1.840	dnf	1.764
1fiy	874	dnf	1.937	1.785	dnf	dnf	1.740	1.830	1.810
PDB	Residues	Linear	Centroid	b = 16	b = 17	b = 18	b = 19	b = 20	DBAve.
1bhe	376	1.832	1.802	1.546	1.421	1.559	1.629	1.581	1.540
1xwl	580	2.036	2.055	1.646	1.738	1.827	1.744	1.886	1.843
1gof	639	1.876	dnf	1.743	1.650	1.661	1.620	1.678	1.732
1t70	2040	dnf	dnf	1.747	1.838	1.822	1.870	1.819	1.804
1aon	8015	dnf	dnf	dnf	dnf	dnf	1.607	1.628	1.622

Table 4: Evaluation of the methods in this chapter on proteins with the common theme of DNA excision repair, where b is the number of bins used in the distance bin method. "dnf" signals that a maximum weight clique was not found within a time limit of 5 minutes. Results are in RMSD Å. "DBAve." is the average result for the distance bin method using 5-20 bins.

PDB	Residues	Linear	Centroid	b=16	b=17	b=18	b=19	b=20	DBAve.
1d4u	111	dnf	dnf	1.293	1.441	1.257	1.507	1.523	1.341
1qze	214	2.358	dnf	1.507	1.467	1.488	1.464	1.427	1.560
1xsn	324	1.976	2.043	1.852	1.709	1.710	1.698	1.779	1.769
2uug	445	dnf	dnf	1.724	1.833	1.823	1.912	dnf	1.787
1xsl	1305	dnf	dnf	1.846	1.812	1.831	1.864	1.860	1.820

3.5 Conclusion

The algorithms presented here offer simple ways to understand and generate solutions to the protein side chain packing problem. They can be implemented relatively quickly, and with good implementations can be very useful for prediction on older computers or devices with a minimum of memory and processor abilities. Many of the proteins evaluated in this study could have their side chains calculated using less than 64 MB of memory. Excluding the clique finding step, which has the demonstrated possibility of failing to find a solution in a given amount of time, the algorithms require no more than N^2R^2 consistency checks, which, as demonstrated via the distance bin method, is often not required.

While this time is not optimal, the algorithm is still useful, particularly for quick evaluation of small proteins with a large number (> 50) of rotamers. This is akin to consider whether to implement insertion sort or quick sort for a list of at most 10 numbers. While quick sort's speed is an asset in the limit, insertion sort's practicality on small datasets cannot be denied, and similarly, the algorithms given in this chapter can be very practical for some proteins, as opposed to some of the more complicated solutions to the SCPP.

Many extensions to this work are possible, including hybridized methods (e.g., first divide by centroid, then divide the resulting centroids by linear or distance bin subpolymers), recursive centroids, inclusion of rotamers to augment the search space, inclusion of disulfide bridges and other physicochemical constraints, or a decomposition method specifying an upper limit to the number of residues in a subpolymer. It would be very interesting to set up a mathematical program to solve for the optimal number of bins in the distance bin method, in hopes of minimizing the overall number of rotamer consistency checks.

Chapter 4 Machine Learning approach - SVM-Based DNA Repair

4.1 Introduction

Though as a race we might not be aware of it, ourselves, our favorite pets, and other living organisms around us are constantly being bombarded by conditions which threaten the stability of our DNA. Whether, in the human case, it be UV radiation (especially the UV-C rays [25]) absorbed while sun tanning or working in the garden all afternoon, secondary tobacco smoke taken in from the neighboring smoking section in a restaurant, or, in the case of mice, the invisible but present electromagnetic field waves generated by cell phones [12], living cells are constantly subject to DNA damage. As mentioned in Chapter 1, it is not only the environment which damages our cells, but also there are a number of fundamental processes in living beings that constantly challenge DNA's stability.

Fortunately, living organisms have evolutionarily developed mechanisms to counteract the effects of the constant damage incurred to DNA. These mechanisms, some of them tied to each other while others are independent, are collectively titled *DNA repair*. DNA repair comes in an assortment of styles, such as repairing the backbone strands holding DNA together or, however inefficient it may be, replacing a single nucleotide base that has been altered or miscopied [25]. A more complete list of repair mechanisms was given in Section 1.1 - *What is DNA Repair?*

DNA repair studies have been ongoing for a considerable length of time [21, 27, 32], and there is a fairly good amount of knowledge available on the topic [20, 25]. Most of these studies have been done in a traditional laboratory under manual curation. However, as the information age has come about, sequence or pathway information for hundreds or thousands of proteins is available in a matter of seconds [24]. These proteins can be tested for a wealth of properties, such as secondary structure [14] or interaction with each other ("protein-protein interaction") [22].

DNA repair has yet to be considerably analyzed from a bioinformatics per-

spective. As a result, this chapter aims to do new research in combining a traditionally observation-based field with modern information processing techniques. Where this work fits into a bigger picture will be discussed in more detail in Chapter 5.

In this chapter, we consider the problems of automatically recognizing and classifying DNA repair proteins. We begin by formalizing our objective and explaining the proper background for the methods enlisted. Next, we cover the types of proteins that are used in this study as well as the necessary technical requirements for enacting the research. An extensive presentation of experimental results follows, and finally we conclude by discussing the implications and knowledge gained from the work.

4.2 Problem Definition

In this work, we consider two problems. First, given a set of proteins, can we accurately identify which of them are related to DNA repair? Second, given that a particular protein is related to DNA repair, can we predict in what capacity it is related, that is, its function or role?

The motivation for this is that if we can establish an accurate method to identify DNA repair proteins, then as new genomes continue to be sequenced, we can quickly and reliably identify that species' DNA repair proteins. This leads to subsequent research and analysis, such as the number of proteins per organism versus its evolutionary history or genome size, or topics to be discussed in Chapter 5.

Formally, for each experiment we are given the three datasets D_+, D_-, and D_0, where D_+ represents a set of known DNA repair proteins, D_- is a set of known proteins that are not related to or involved with DNA repair, and D_0 contains the set of proteins on which we wish to decide for each protein in the set whether or not it is a DNA repair-related protein. Each dataset D contains a finite number of proteins p_1, p_2, ..., $p_{|D|}$, initially given in terms of their amino acid or nucleotide sequence information. After transforming this sequence information into an appropriate format to be described in Section 4.2.2, we train a machine learning algorithm called a Support Vector Machine

(hereafter called SVM, see Sections 2.3.2 and 4.3.4) to learn from the sets D_+ and D_-, creating a criteria (mathematical representation) which can be used on a protein whose status is unknown in order to predict whether or not it is a DNA repair-related protein. We perform this process for a logical progression of datasets and analyze the results.

After the first problem of recognition of DNA repair proteins has been addressed, we continue on to the problem of classifying proteins that are known to be DNA repair-related. For this problem, D_+ and D_- are sets of differing types of DNA repair proteins. Creation of D_0 for both the recognition and classification problems is discussed below.

4.2.1 Separable Datasets

In each of the experiments done (see the progression given in Section 4.3), we perform 2- through 10-fold cross validation. In k-fold cross validation, we divide the sets D_+ and D_- into k proper subsets (formally $D_+ = D_{+1} \bigcup D_{+2} \bigcup \ldots \bigcup D_{+k}$ and for $p_i \in D_{+a}, p_i \notin D_{+b}, a \neq b$, where $D_{+a} \subset D_+$, and analogously for D_-). We combine $k-1$ subsets of D_+ and D_- as training data, and combine the remaining one subset from each of D_+ and D_- for testing data. We then repeat this process k times, each time leaving out a different lone subset from each of the larger datasets.

In the classification experiment, we classify a set of DNA repair proteins into one of six classes: excision repair, mismatch repair, recombination repair, repair helicases, repair nucleases, or other general repair. For each type of repair, we consider it to be the positive class D_+ and leave the remaining five classes to be the negative sample set D_-. This is called one-versus-rest (1-v-r) validation. Here we also use cross-validation in assessing the classification accuracy.

In each of the cross-validations, we want to create a decision hyperplane that we can use to classify the test samples. This is achieved by using a SVM, which seeks to create the maximum width boundary between the two training sets. After creating our decision boundary (an example was given in Section 2.3.2, Figure 10), each test protein input will fall on one side of the boundary, thereby achieving a prediction as to whether or not it is a DNA repair-related protein.

Table 5: The input dimensionality of vectors for experiments performed in this chapter, when using amino acids for primary sequence information. k_1 is the length for the spectrum kernel that processes the primary sequence information and k_2 represents the length of the spectrum kernel that processes the secondary sequence information. Underlined values indicate that the secondary structure kernel contributes more to the overall dimensionality of the resulting vector.

k_2	$\lvert\Sigma_2^{k_2}\rvert$	$k_1 = 1$	$k_1 = 2$	$k_1 = 3$
0	0	20	400	8,000
1	3	23	403	8,003
2	9	29	409	8,009
3	27	47	427	8,027
4	81	101	481	8,081
5	243	263	643	8,243
6	729	749	1,129	8,729
7	2,187	2,207	2,587	10,187
8	6,561	6,581	6,961	14,561

4.2.2 Kernel Calculations

As stated in Section 4.2, each protein p_i is initially described using its amino acid or nucleotide sequence information (see Section 2.1.1 for a list of amino acids and nucleotides). However, since SVMs use numerical vector data [15, 23, 33], we must transform the sequence data into some sort of numerical representation. This is achieved using the *spectrum kernel*.

The spectrum kernel [33] is a technique which counts the number of occurrences of each key in a set against a query object. As a more concrete example, if we are dealing with sequences, have the keys $\Sigma = \{A, B\}$, and we query against the sequence $S = ABBAB$, then the spectrum kernel $K_s(\Sigma, S)$ will return to us the vector

$$K_s(\{A, B\}, ABBAB) = (2, 3)$$

because there are two As and three Bs.

More generally, if we consider the *k-spectrum kernel*, we can consider all com-

binations of exactly k input symbols $\sigma \in \Sigma$, denoted by Σ^k. The meaning of K_s remains the same: $K_s(\Sigma^k, S)$ is the $|\Sigma|^k$-length vector representing the number of occurrences of each possible sequence $\sigma_a\sigma_b \ldots \sigma_k$ for $\sigma_a, \ldots, \sigma_k \in \Sigma$ that is exactly k input symbols long. To extend the previous example, consider the 2-spectrum kernel Σ^2 for $\Sigma = \{A, B\}$. We now have $\Sigma^2 = \{AA, AB, BA, BB\}$, since this represents all possible combinations of inputs of exactly length 2. The calculation of K_s is then easily verified upon manual inspection or computation to be

$$K_s(\Sigma^2 = \{AA, AB, BA, BB\}, ABBAB) = (0, 2, 1, 1).$$

Rather than use only the primary sequence information of DNA repair proteins, when dealing with amino acid sequences, we also utilize secondary structure information. For this purpose, we use the secondary structure prediction program SSpro developed at the University of California, Irvine [14]. SSpro predicts whether each amino acid used is part of an α-helix (H), a β-strand (E), or another structural function (C). Since SSPro returns a sequence equal in length to the primary sequence input, we can also perform a k-spectrum kernel on the secondary structure information.

Depending on the type of primary sequence data used, our spectrum kernel will return one of several types of sequences. For nucleotide sequences, $|\Sigma| = 4$ and hence we transform input protein p_i into the form

$$K_s(\Sigma^k, p_i), k = \{1, 2, 3\},$$

giving us output vectors of lengths 4 ($k = 1$), 16 ($k = 2$), or 64 ($k = 3$), where $k = 3$ captures the biological sense of a codon. For amino acids, we can choose to include or exclude secondary structure data. This gives us two different sets of spectrum kernel symbols: $|\Sigma_1| = 20$ for primary sequence information, and $|\Sigma_2| = |\{C, E, H\}| = 3$ for predicted secondary structure data. We denote the resulting spectrum kernel by $K_s^{'}$. Computation using $K_s^{'}$ produces vectors of the form

$$K_s^{'}(\Sigma_1^{k_1}, \Sigma_2^{k_2}, p_i), k_1 = \{1, 2, 3\}, k_2 = \{0, 1, 2, \ldots, 8\},$$

giving us output vectors ranging in length from 20 to 14,561 (by enumerating

all k_1 and k_2, see Table 5).

Note that the resulting vector length will become the dimensionality of the hyperplane (separation criteria) we wish to find. As can be imagined, vector calculations in large dimensions (such as 1,000 or more) is not an easy task. Fortunately, there are implementations of the SVM technique that can nonetheless handle this dimensionality, and we discuss one of them in the next section.

4.3 Materials and Methods

In this research, we introduce a natural progression of experiments designed to observe the effect of changing one dataset at a time. In total, we use seven different datasets; three DNA repair-related datasets and four non-repair datasets are combined as shown in Table 6. How each of these datasets is obtained is described in the sections below. As mentioned in Section 4.2.1, we perform cross validation on each of the positive (D_+) and negative (D_-) datasets for each experiment. Furthermore, all cross validations are performed using the appropriate spectrum kernels described in Section 4.2.2.

After datasets are properly prepared by cross-validation and spectrum kernels, we use an implementation of the Support Vector Machine called SVM^{light} [23] to perform training and testing. Details for SVM^{light} can be found below in Section 4.3.4. Finally, results are systematically collected and analyzed.

4.3.1 DNA Repair Proteins

Our DNA repair protein datasets come from two locations. KEGG [24] is a comprehensive bioinformatics resource, updated daily with new sequence, pathway, interaction, and reaction data, to list only a few available data types. From KEGG, we queried the GENES database with the phrase "DNA repair", and subsequently downloaded the resulting proteins listed using the KEGG API available at the database website. KEGG offers both amino acid and nucleotide representations of proteins, and we utilize both in our experiments. Downloaded proteins are listed in FASTA format.

The UniProt [7] database provides sequence, functional, classification, or property information for proteins. Using a web form, we queried the UniProtKB database sequence field for "DNA repair." Results are available in XML,

Table 6: The series of DNA repair protein recognition experiments performed. At each step we change only one dataset in order to assess the usefulness of each dataset. Numbers under the size columns indicate the number of unique proteins, with respect to primary sequence, extracted from the respective databases. NUC signals nucleotide sequence input, while AA indicates amino acid sequence (and possibly secondary structure) input.

Exp. No.	Input	Positive	Size	Negative	Size
1	NUC	KEGG DNA Repair	3374	KEGG Histone	2118
2	AA	KEGG DNA Repair	3374	KEGG Histone	2118
3	AA	KEGG DNA Repair	3374	NIH Histone	1367
4	AA	KEGG DNA Repair	3374	UniProt non-repair	4358
5	AA	UniProt DNA Repair	2936	UniProt non-repair	4358
6	AA	UniProt DNA Repair	2936	NIH Histone	1367

FASTA, or table format, or can be downloaded as a flat file.

4.3.2 Histones

Histones are a type of protein which are found in chromatin (the material of DNA including its backbone and nucleotides). They bind to DNA to help enforce its natural helical structure, and there are five primary categories [11, 28]. Hence, this binding means that they are very similar to DNA repair proteins, and serve as a good dataset for testing distinguishment between repair and non-repair proteins.

The KEGG database [24] has a collection of histones available for download. As in Section 4.3.1, we use the KEGG API to download histones available from this database.

The National Institutes of Health's National Human Genome Research Institute also maintains a database of histones [28], drawing from several large databases. The database uses BLAST sequence alignment to create a highly specific database that removes redundancy as much as possible. Bhasin *et. al* used basic amino acid and dipeptide composition spectrum kernels in conjunction with this database to identify histones [11]. NIH histone sequences are

presented in FASTA format, and proteins from this database were copied via a web browser into a plain text file.

4.3.3 Other Data Categories

As the non-repair datasets D_- mentioned so far have been only histones, it is logical to test how well the methods here perform with a dataset other than histones. For this purpose, we again use the UniProt [7] database. Rather than search for solely another type of protein other than histones, we instead query for a more complete dataset encompassing proteins with many functions.

By querying for "nuclear NOT dna repair", UniProt returned to us a database of proteins that are involved in activities in the nucleus but not related to DNA repair. Since DNA repair occurs in the nucleus, this large database of proteins that are also in the nucleus but not repair-related is a useful database to assess the computational methods proposed.

4.3.4 SVM^{light}

It is unnecessary to manually implement the SVM algorithm. Joachims has developed the SVM^{light} software package which efficiently implements the SVM algorithm [23] and is available for research use. Amongst the many functions provided by SVM^{light}, binary classification and regression are two of the most often performed tasks.

SVM^{light} is used for our spectrum kernel technique, as it requires simply inputting each feature number and frequency. Positive or negative membership is specified for each training sample. For example, if we refer to the 1-spectrum example in Section 4.2.2 using $\Sigma = \{A, B\}$ and assuming our input protein $p_i = ABBAB$ is a negative training sample, then usage of SVM^{light} is simply to input

$$< -1, 1 : 2, 2 : 3 >,$$

where $1 : 2$ means that the first feature, the number of As, is 2 and the second feature, the number of Bs, is 3.

In our experiments, we normalize all inputs so that their values range from 0 to 1. In doing so, our example will actually be input as

$$< -1, 1 : 0.4, 2 : 0.6 > .$$

As mentioned in Section 4.2.2, our input vectors ranged in length from 4 to 14,561.

4.4 Results

We performed a total of 138 recognition experiments (six experiments, three primary sequence spectrum kernels per experiment in addition to five experiments with three primary sequence spectrum kernels combined with one of eight secondary structure spectrum kernels) on the datasets given in Table 6. Though it is impossible to show the result of all 138 experiments, each 2- through 10-fold cross-validated, we show a few of the experiments for which we can clearly identify patterns in the experimental results. We summarize the results of the two most biologically realistic experiments, Experiments 4 and 5, into the single Table 7 for quick reference, with more comprehensive analysis provided via figures in Section 4.4.1.

Based on the work of Bhasin *et. al* which showed superior SVM performance over BLAST for identification and classification amongst a general group of proteins [11], we did not perform comparative BLAST experiments for the protein datasets used in this thesis.

Before showing the results, we note here that in Figures 18 - 33, "linear" signals a linear kernel used for similarity in SVM^{light}, "poly-5" is a 5th degree polynomial kernel, "poly-10" is a 10th degree polynomial kernel, and "rbf-G" is a radial basis function (RBF) kernel with parameter $\gamma = G$ (see [15, 23, 33] for an explanation of the γ parameter).

In addition to the recognition experiment, we also give the results of a classification experiment using DNA repair proteins found in KEGG [24] as well as show a public web server created for the DNA repair community.

4.4.1 Experimental Results

Primary Structure Prediction Accuracy For our recognition experiments, we first examined the performance of the SVM when using only primary structure sequence data, as well as how this performance is affected by the primary structure kernel K_s described in Section 4.2.2. We begin by using the nucleotide sequences of DNA repair proteins and histones in KEGG. Results of 1-, 2-, and

Table 7: A brief summary of key experimental results, in accuracy (%), obtained using Support Vector Machines with multiple spectrum and similarity kernels for recognition of DNA repair proteins. As throughout this chapter, k_1 indicates a k_1-spectrum primary structure sequence kernel and k_2 indicates a k_2-spectrum secondary structure sequence kernel. The numerical values shown under each experiment column are the average accuracies obtained per experiment with a corresponding combination of k_1 and k_2, using a 10th degree polynomial kernel and a $\gamma = 2.4$ RBF kernel validated via 2- through 10-fold cross validation.

Primary k_1	Secondary k_2	Experiment 4		Experiment 5	
		poly-10	rbf-2.4	poly-10	rbf-2.4
1	0	82.84	82.82	82.12	82.03
1	1	86.80	86.82	85.99	85.77
1	8	87.58	87.86	87.19	87.44
2	0	84.62	85.85	84.77	85.48
2	1	86.47	86.16	85.88	85.61
2	8	88.06	89.37	87.76	88.68
3	0	88.05	89.05	87.12	87.68
3	1	84.25	84.64	83.79	84.02
3	8	86.43	88.28	87.03	87.84

3-spectrum kernels are shown in Figure 18 by measuring the average accuracy across a k-fold cross-validation test. It is apparent from the figures that as the spectrum kernel level increases, the prediction accuracy increases. The increase in dimensionality helps the SVM algorithm better distinguish between repair-related proteins and non-repair proteins. An average accuracy of 82% is achieved using the 3-spectrum kernel on nucleotide sequences, which is good for machine learning experiments.

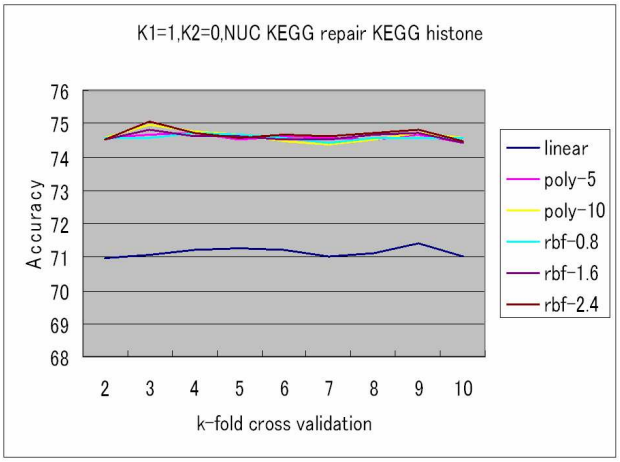
The next experiment done was to test the difference in prediction accuracy by transitioning from nucleotide sequence data to amino acid sequence data for the same repair- and histone-related proteins. Using a 3-spectrum kernel on nucleotide data captures the sense of a codon, which encodes for an amino acid. We might then expect that the 3-spectrum kernel on nucleotide sequence data will produce the same results as a 1-spectrum kernel on amino acid sequence data.

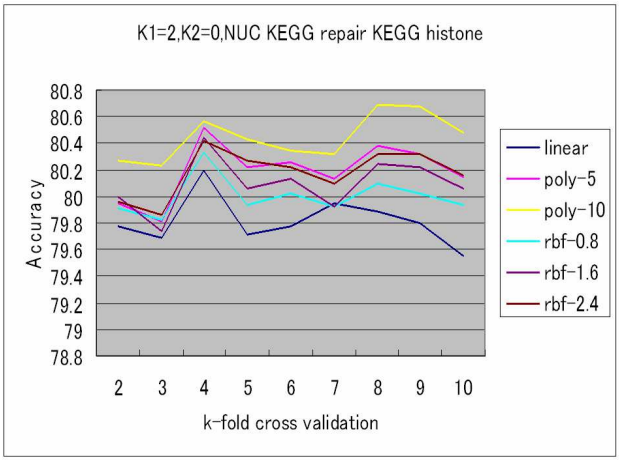
As shown in Figure 19, the use of 1-spectrum kernel amino acid sequence data outperforms the nucleotide spectrum kernels at all levels. We attribute this to several factors, such as the variety of input types and assumption of correct codon reading frames in the amino acid sequence data. The improvement in using amino acid data is considerable, going from 71% using 1-spectrum nucleotide sequence kernels to approximately 94% when using 3-spectrum amino acid sequence kernels.

As a result of this difference, we find it unnecessary to evaluate our datasets with both nucleotide and amino acid sequence data in the remainder of experiments performed. Experiments 2, 3, 4, 5, and 6 are performed using only the amino acid representation of repair and non-repair proteins.

In our third experiment, we replace the KEGG histone dataset with the histone dataset available from the National Institutes of Health [28]. The NIH histone dataset has been shown to achieve high recognition accuracy in other experiments [11], and to our surprise, it achieves an accuracy of more than 95% in our experiments. As Figure 20 shows, 99% recognition accuracy occurs even when using a 1-spectrum primary sequence kernel.

Our first three experiments used histones to act as a negative dataset. Since





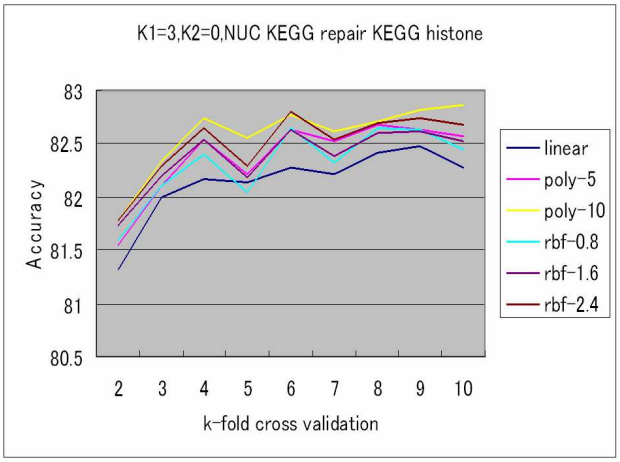
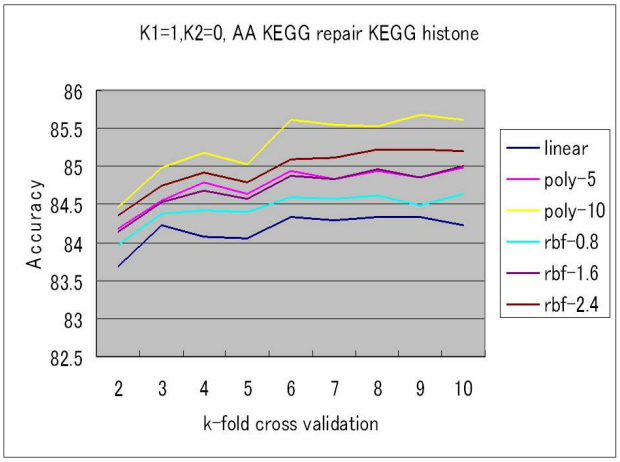
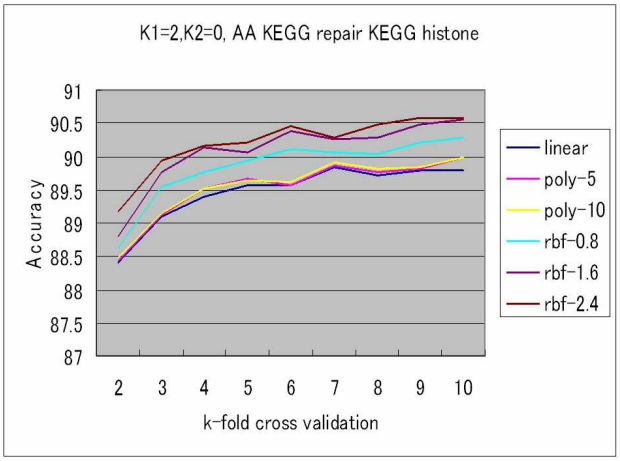


Figure 18: Results of 1-, 2-, and 3-spectrum nucleotide primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from KEGG as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.





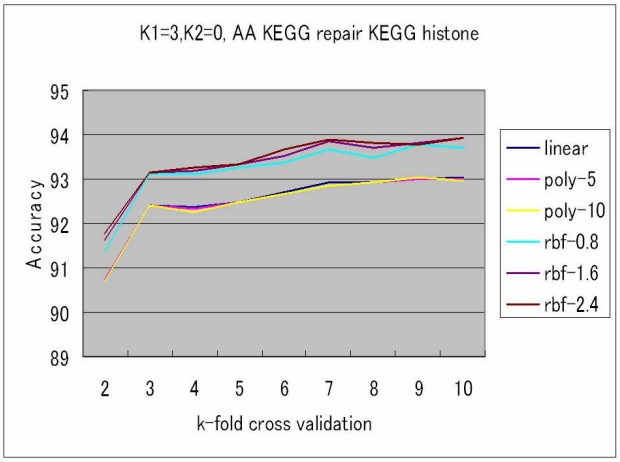
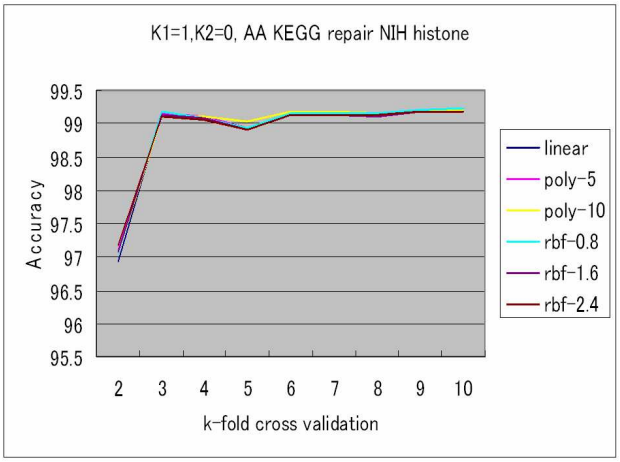
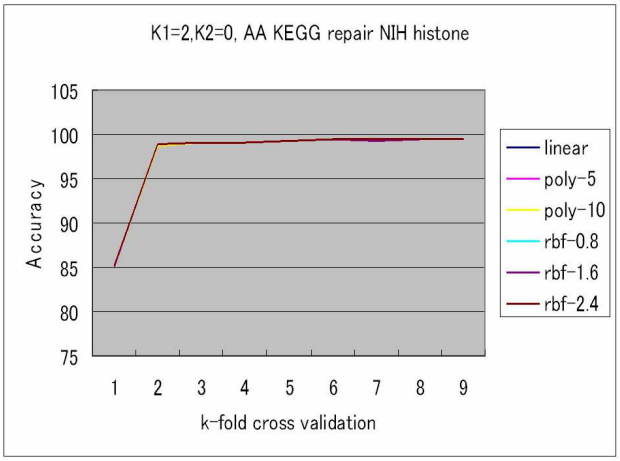


Figure 19: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from KEGG as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.





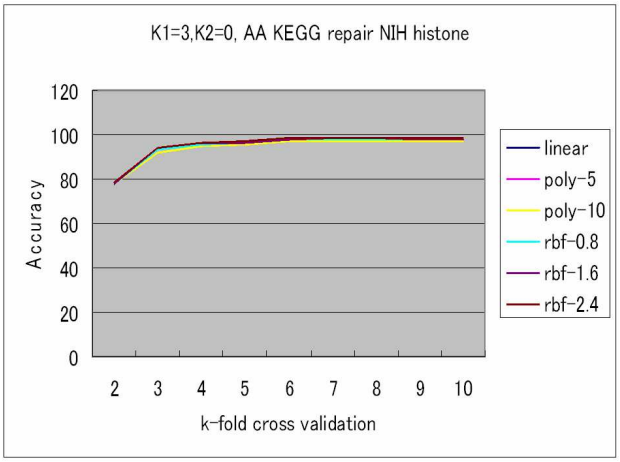


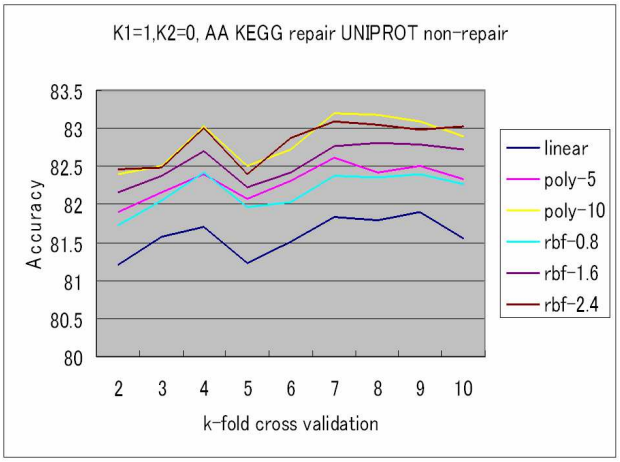
Figure 20: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from NIH as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.

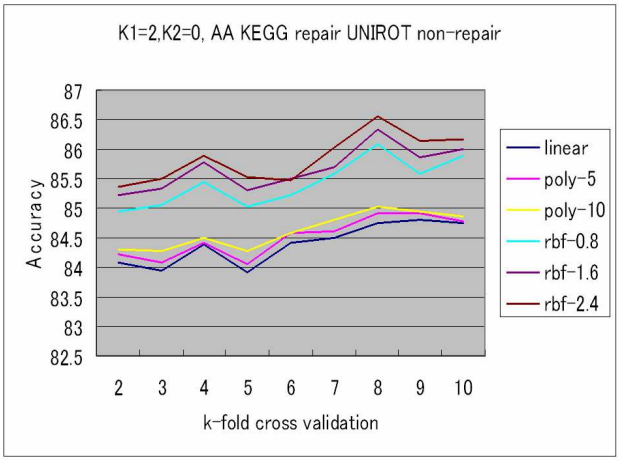
there are many more types of proteins besides histones that are not involved in DNA repair, it is logical to introduce a negative dataset that has a larger scope than only histones. As explained in Section 4.3.3, we obtained a set of 4,358 non-repair-related nuclear proteins from the UniProt [7] database. Our experimentation using this dataset gives us a more realistic impression of the overall performance of our computational techniques. Accuracy from 81-90% is achieved, again giving us an approximately 10% performance boost by using a 3-spectrum kernel instead of a 1-spectrum kernel, similar to Experiment 2 between KEGG DNA repair proteins and KEGG histones. The results of the experiment using KEGG DNA repair and UniProt nuclear non-repair proteins are shown in Figure 21.

Following the scientific method of changing one variable at a time, our next experiment is to test our framework using a positive dataset other than KEGG DNA repair proteins. For this purpose, we again utilize the UniProt database [7], querying for DNA repair proteins. Our positive dataset of 2,936 proteins is slightly smaller than the KEGG repair protein dataset, but nonetheless we achieve 80-89% prediction accuracy, as shown in Figure 22. Though the result is slightly less than the 84-94% performance range of Experiment 2, the datasets used in Experiment 5 (the experiment given in this paragraph) should be considered to be more useful in a realistic analysis of the computational techniques proposed, because both the positive and negative datasets are representative of many organisms and many protein functions.

Finally, we perform one additional experiment to further analyze the unexpectedly high performance results from use of the NIH histone database. This experiment replaces the KEGG DNA repair protein dataset with the UniProt repair dataset, and again uses the NIH histone database as negative example data. The performance results in Figure 23 are similar to Experiment 3, achieving 99% accuracy for primary sequence spectrum kernels of all three levels.

Adding Simple Secondary Structure Information Primary structure sequence information alone achieves fairly good performance using the methods described here. However, secondary structure sequence prediction programs can be useful in offering even more information for the Support Vector Machine





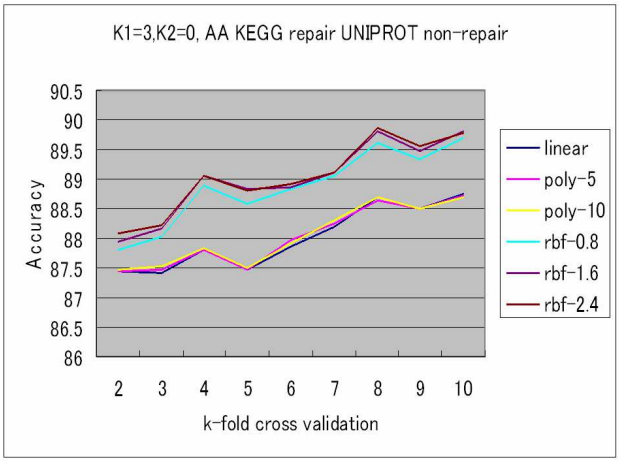
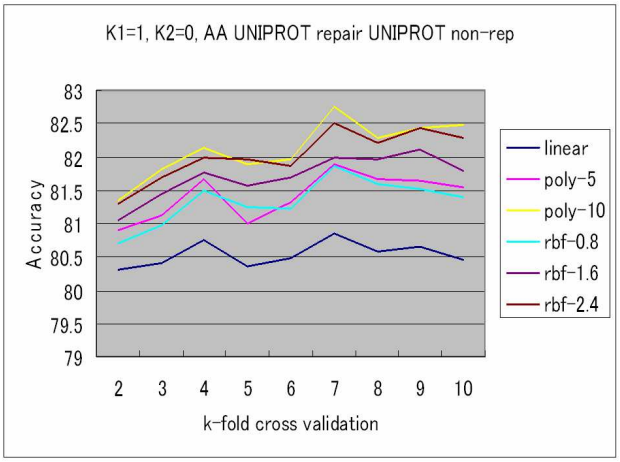
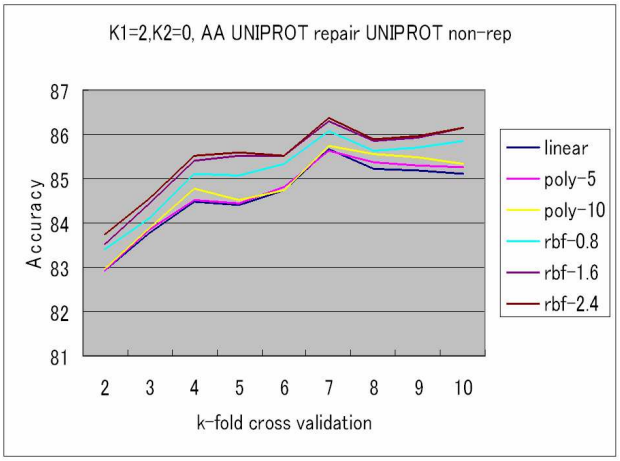


Figure 21: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.





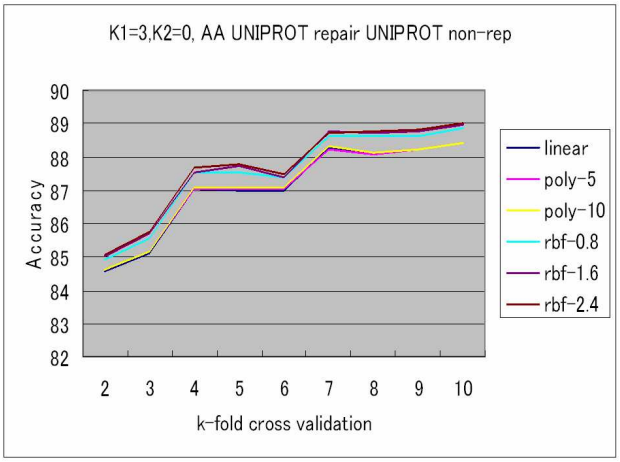
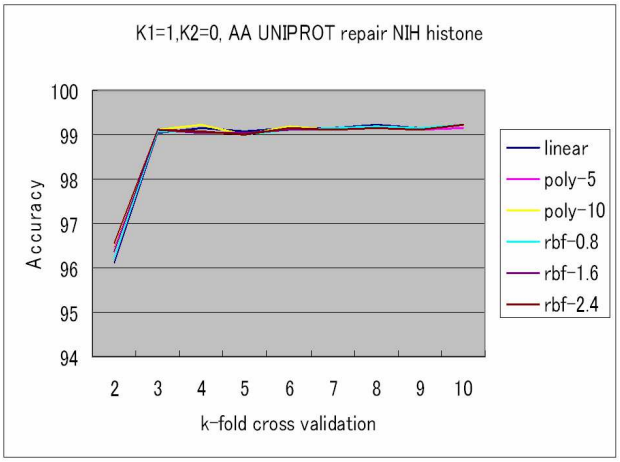
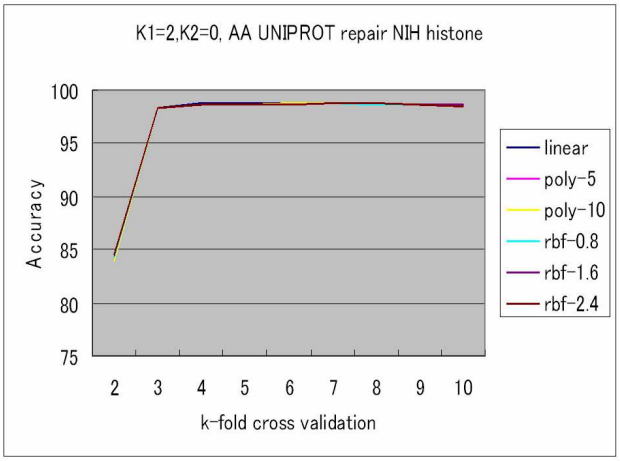


Figure 22: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.





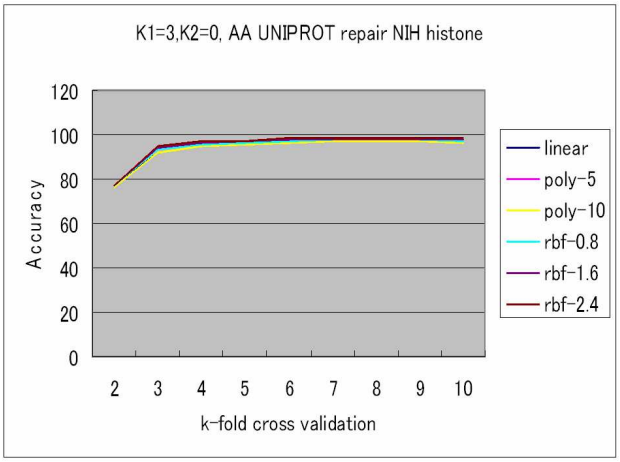


Figure 23: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and histones from NIH as negative data. K1 signals the primary sequence spectrum kernel level, and K2=0 indicates that no secondary structure sequence information is used.

method to create a decision hyperplane. We performed the same five experiments on amino acid sequence datasets, but introduced the addition of three more parameters measuring the frequency of α-helices, β sheets, and acids with other structural roles. Here, we note the major effects observed from experimental results (Figures 24 - 28) achieved using these three extra parameters (hence k_2 described in Section 4.2.2 has a value of 1).

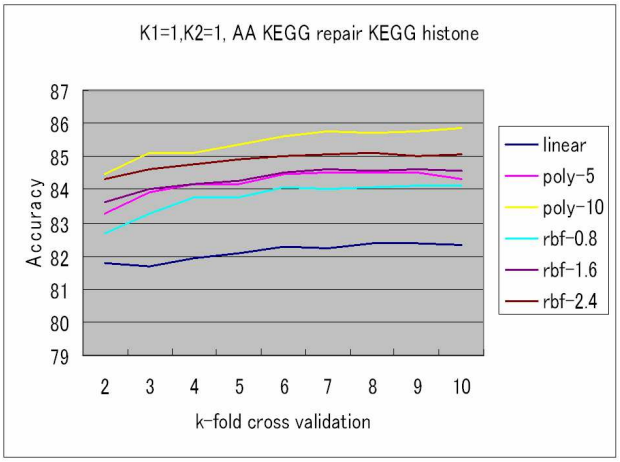
In nearly all situations, the inclusion of secondary structure information for $k_2 = 1$ had an adverse effect. This is not necessarily a drawback, but rather a confirmation that adding three considerably ambiguous dimensions to a vector that is vastly larger does not provide a better separation criteria.

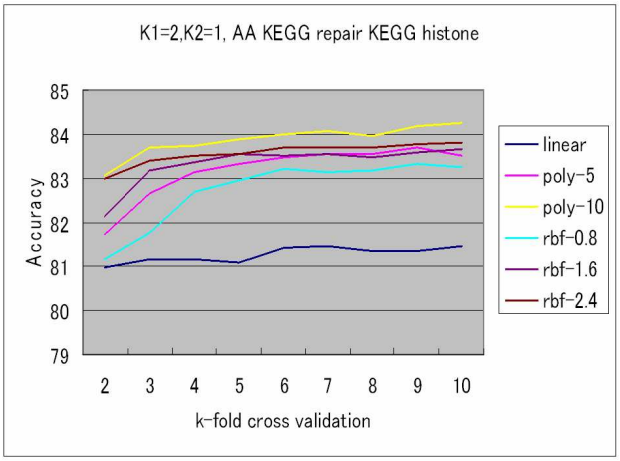
What we do note as a positive is that when we use a 1-spectrum primary sequence kernel, the less complex polynomial kernel does manage to be one of the top performers when simple secondary structure information is added, for example achieving a 1-3% increase in performance over spectrum kernels that do not use secondary structure information when evaluating KEGG DNA repair proteins and KEGG histones.

Experiments 3 and 6 no longer achieve 99% recognition accuracy but are slightly reduced to 95-98% accuracy when simple (again $k_2 = 1$) secondary structure information is added. Here again the polynomial kernel is a top performer.

Having seen that adding only the frequencies of α-helices, β sheets, and acids with other structural roles does not improve our performance, we consider whether or not to discard secondary structure information altogether. As it will turn out, secondary structure information can improve our recognition performance, given a more useful secondary structure sequence spectrum kernel.

Effective Secondary Structure Information Usage In order to make effective use of secondary structure information that is available, it is necessary to use a larger secondary structure sequence spectrum kernel. As was shown in Table 6, $k_2 = 1$ produced output vectors of sizes 23, 403, and 8003. Here, we discuss the results achieved when setting k_2 to a value of 8, which produces vectors of 6581, 6961, and 14561 dimensions. By doing this, the SVM has considerably more dimensionality to work with when seeking the optimal





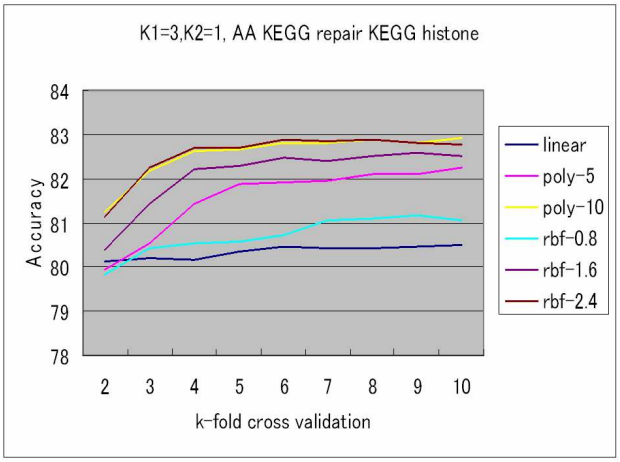
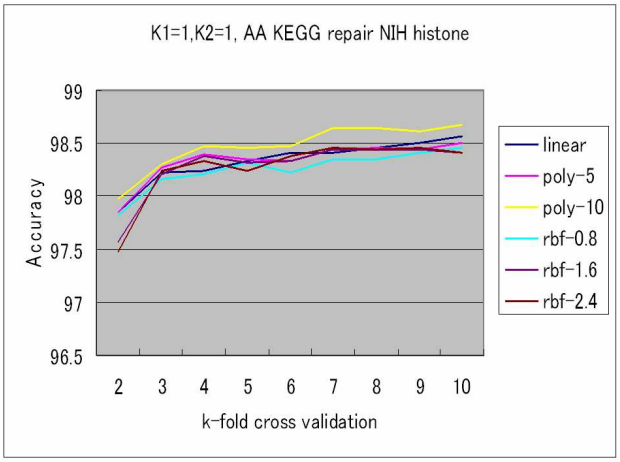
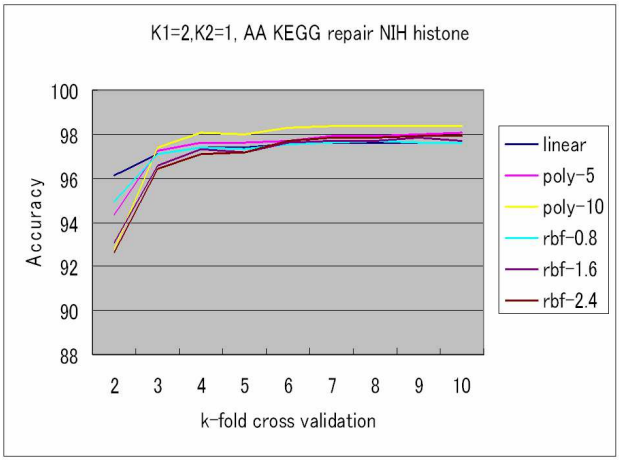


Figure 24: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from KEGG as negative data. K1 signals the primary sequence spectrum kernel level, and K2=1 indicates that we have included α-helix, β sheet, and other structural role frequencies.





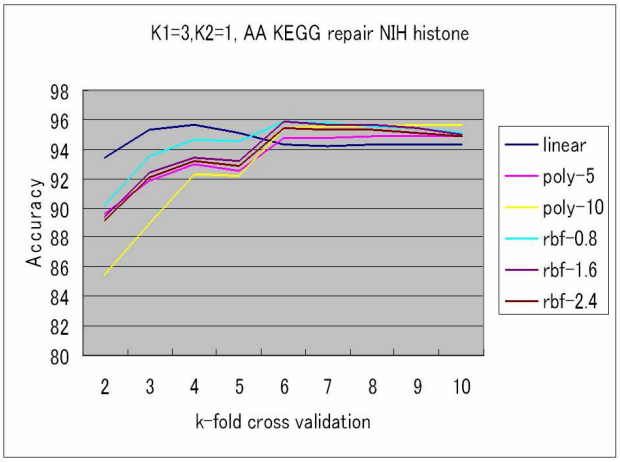
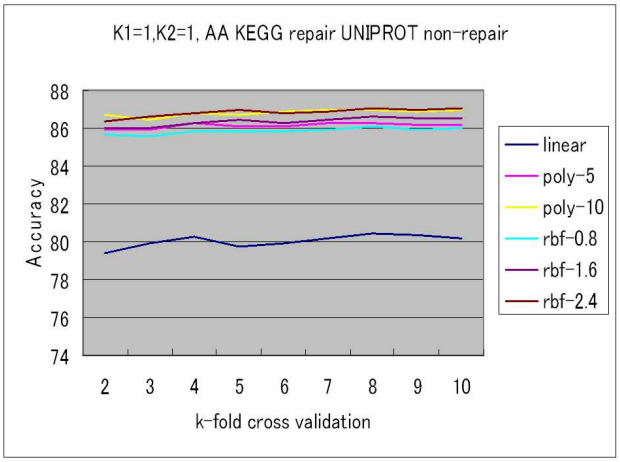
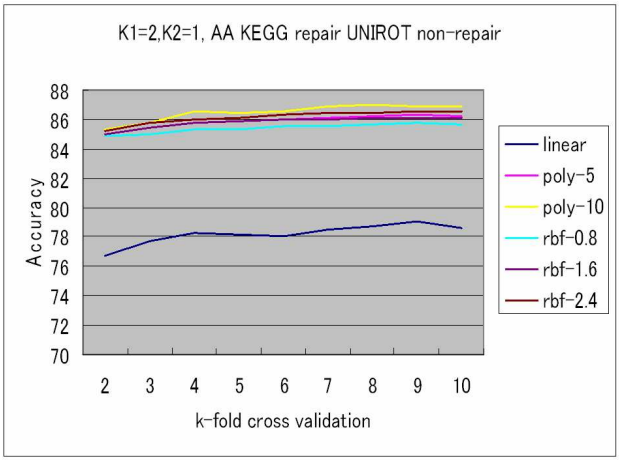


Figure 25: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from NIH as negative data. K1 signals the primary sequence spectrum kernel level, and K2=1 indicates that we have included α-helix, β sheet, and other structural role frequencies.





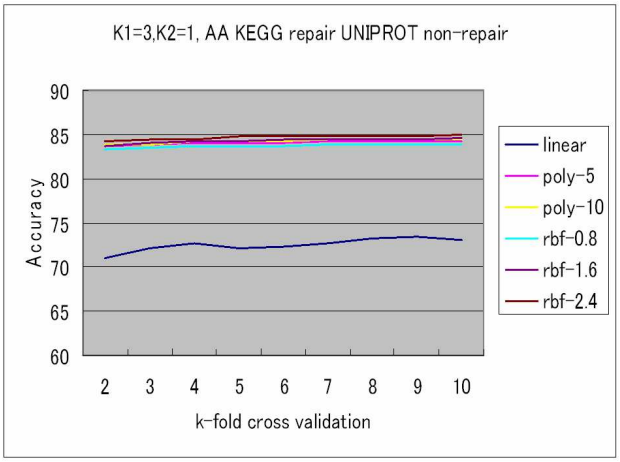
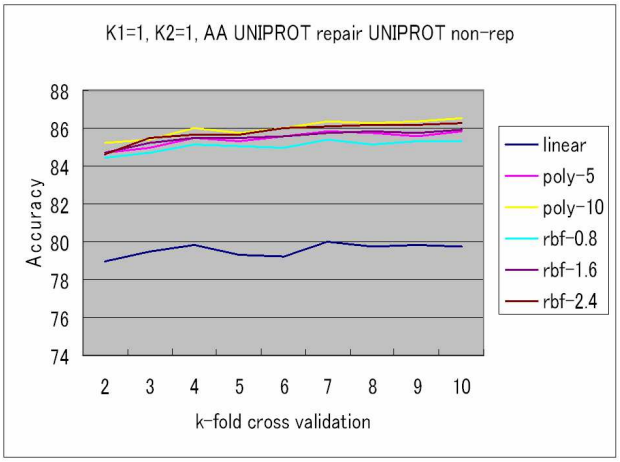
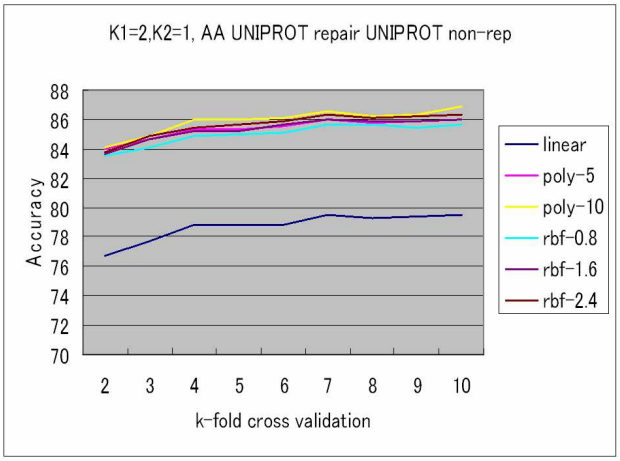


Figure 26: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K1 signals the primary sequence spectrum kernel level, and K2=1 indicates that we have included α-helix, β sheet, and other structural role frequencies.





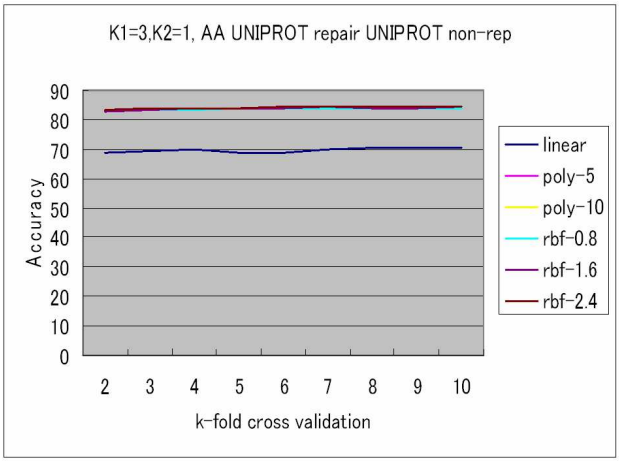
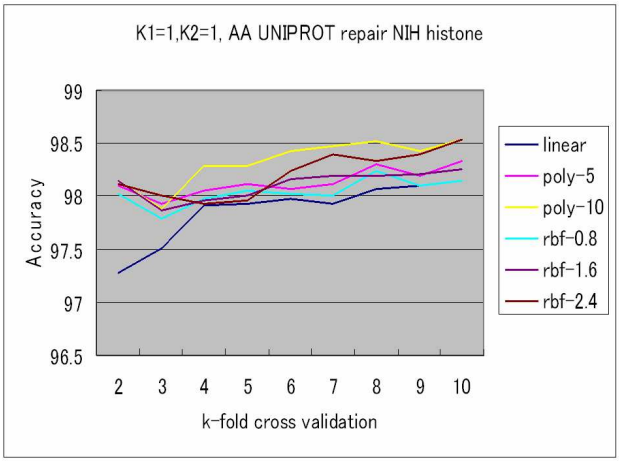
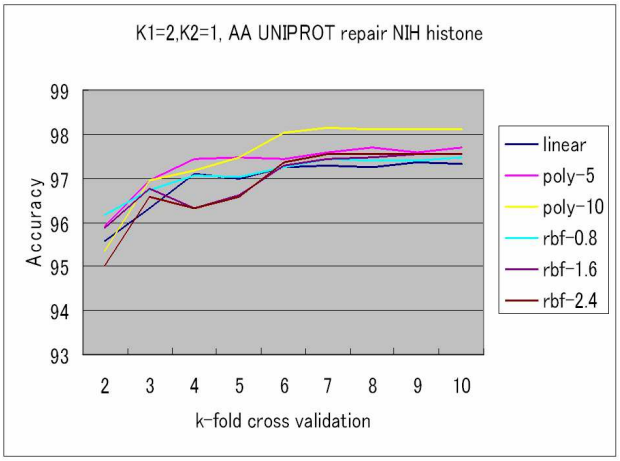


Figure 27: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K1 signals the primary sequence spectrum kernel level, and K2=1 indicates that we have included α-helix, β sheet, and other structural role frequencies.





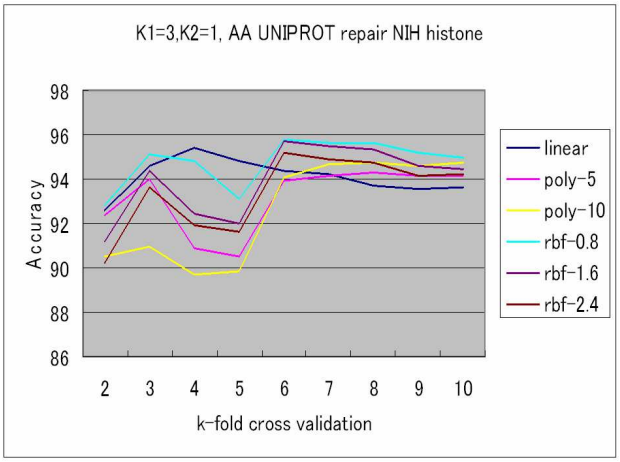


Figure 28: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and histones from NIH as negative data. K1 signals the primary sequence spectrum kernel level, and K2=1 indicates that we have included α-helix, β sheet, and other structural role frequencies.

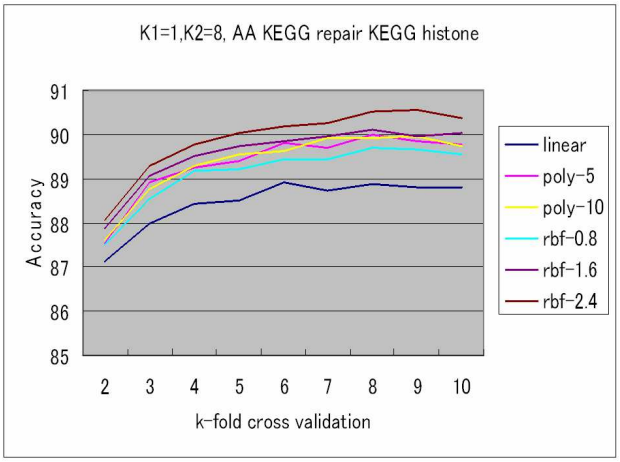
hyperplane for distinguishing between repair proteins and non-repair proteins.

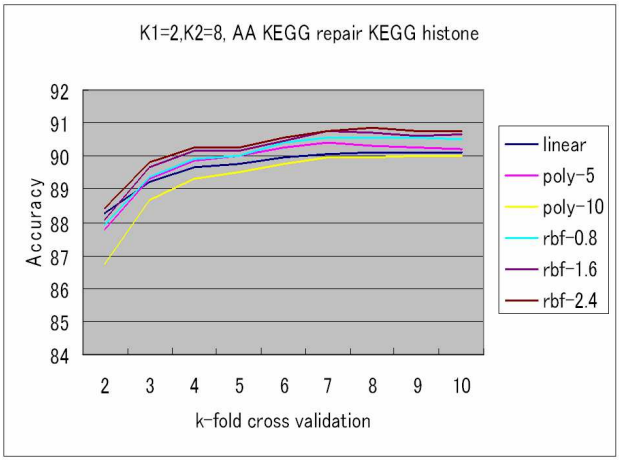
The inclusion of extra dimensionality is a fair boost in our results, providing an extra 5-6% of accuracy when considering KEGG repair proteins and KEGG histones for a 1-spectrum primary sequence kernel. Similar results follow for experiments using the UniProt repair protein and nuclear non-repair protein datasets. Performance in Experiments 3 and 6 deteriorated to 95-98% when using the simple secondary structure sequence spectrum kernel, but performance returns to approximately 99% when $k_2 = 8$.

Clearly visible in Figures 29 - 33, the RBF kernels provide the best performance after the increase in vector dimensionality. The RBF kernel is well detailed in [15, 33], but in short it uses the difference of two vectors to measure their similarity (smaller difference signals more similarity and hence a higher kernel score), and it is this difference that makes the RBF kernel capable of using the extra information of higher dimensions, as opposed to polynomial kernels which may have inner product values of zero and give us little extra information.

Classification Since our recognition experiments perform well, we investigated to what extent the same computational technique could classify DNA repair proteins. Using the DNA repair protein dataset extracted from KEGG we created twelve new datasets, where proteins related to excision repair, mismatch repair, recombination repair, repair helicases, repair nucleases, and other general repair were made into individual positive datasets, with the respective remaining five types of repair designated as negative datasets. This is also known as one-versus-rest (1-v-r) testing.

For each type of 1-v-r testing, we performed 5- and 10-fold cross validation using 1- and 2-spectrum primary structure sequence kernels. We found that the smaller datasets were more successful in classification, explained by the fact that in those cases there are larger numbers of negative training examples to build a better criteria for classification. For datasets that did not achieve a 90% or higher accuracy rate when using 1-spectrum kernels, they generally showed a 10% improvement in accuracy after the transition to the higher-dimensional 2-spectrum kernel. This concurs with our analysis for the recognition experiments,





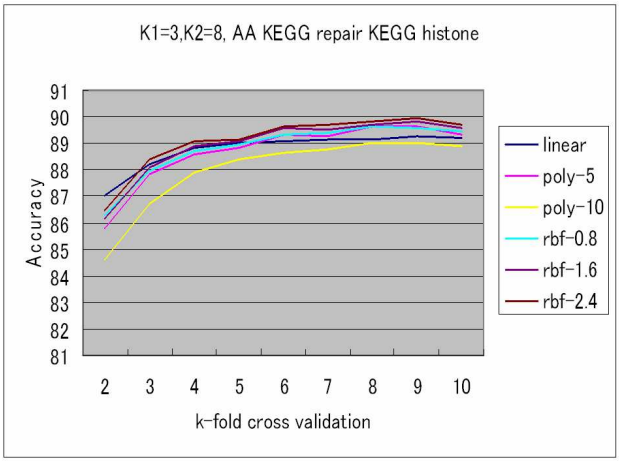
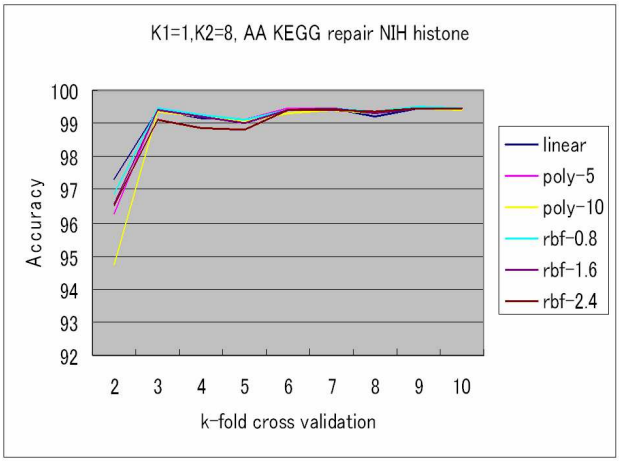
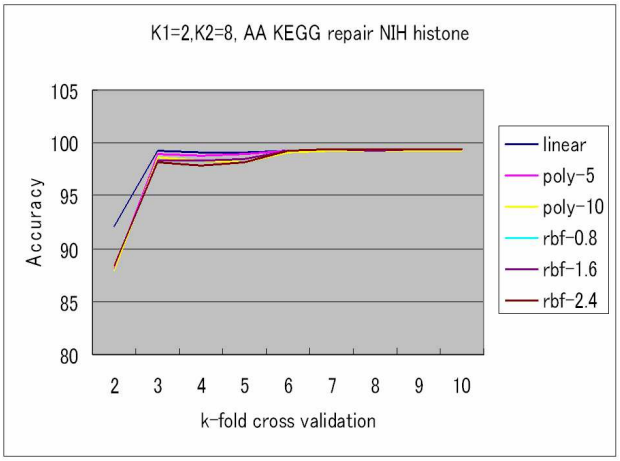


Figure 29: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from KEGG as negative data. K2=8 indicates that we have added over 6,000 dimensions of information measuring α-helix, β sheet, and other structural role sequence patterns.





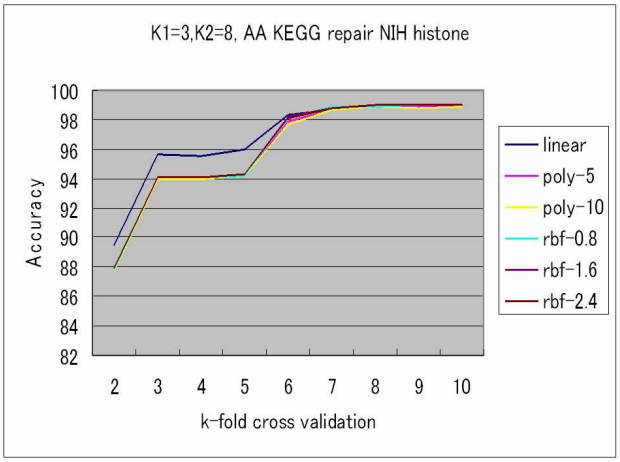
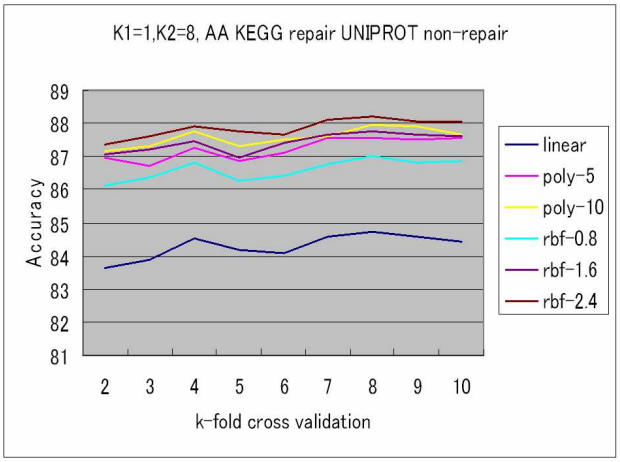
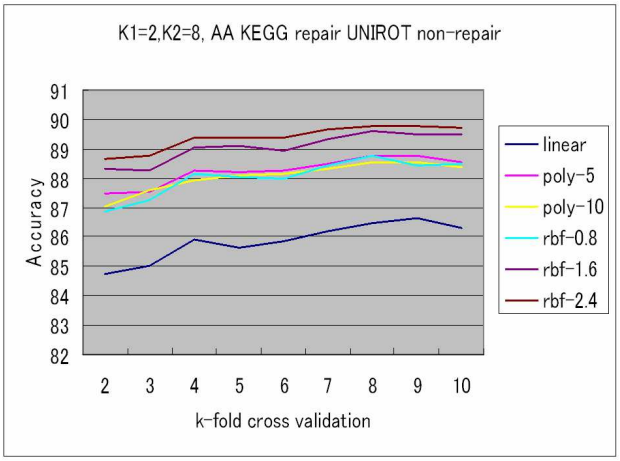


Figure 30: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and histones from NIH as negative data. K2=8 indicates that we have added over 6,000 dimensions of information measuring α-helix, β sheet, and other structural role sequence patterns.





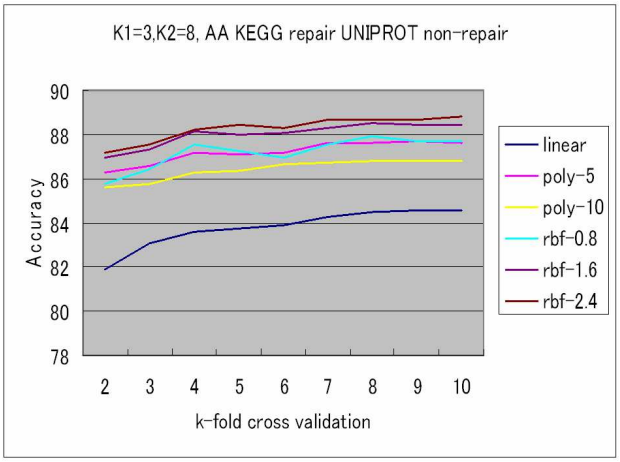
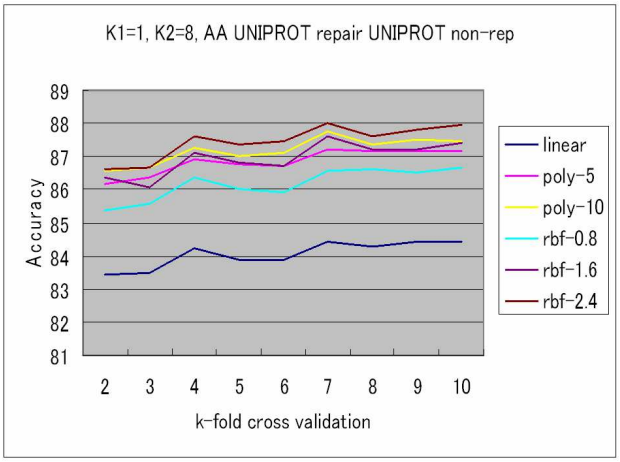
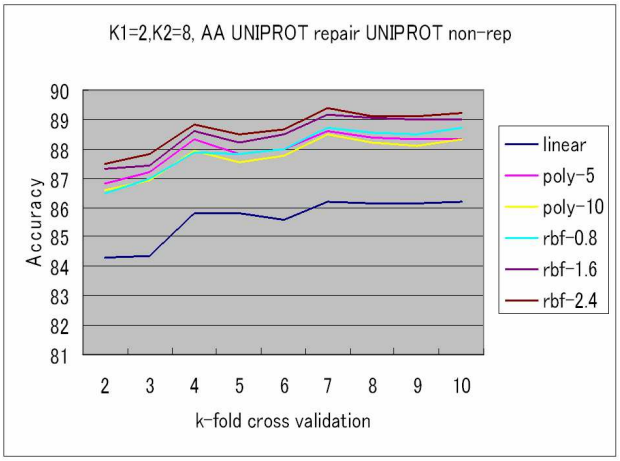


Figure 31: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from KEGG as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K2=8 indicates that we have added over 6,000 dimensions of information measuring α-helix, β sheet, and other structural role sequence patterns.





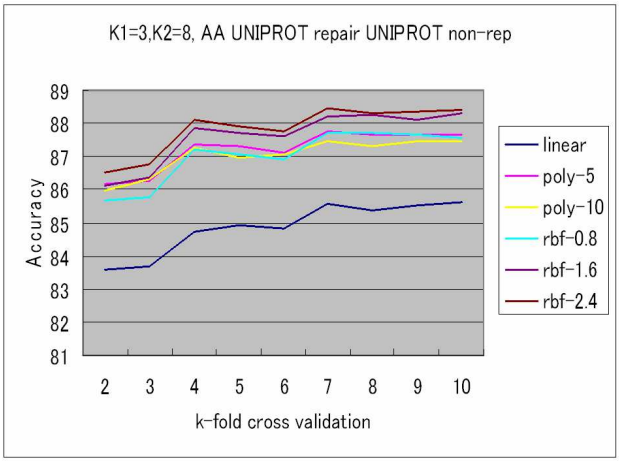
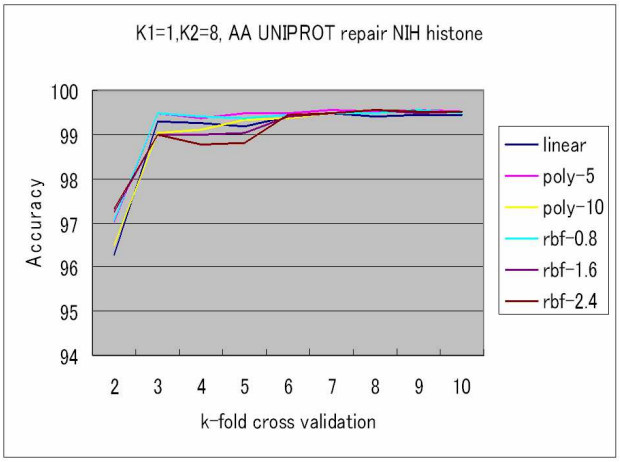
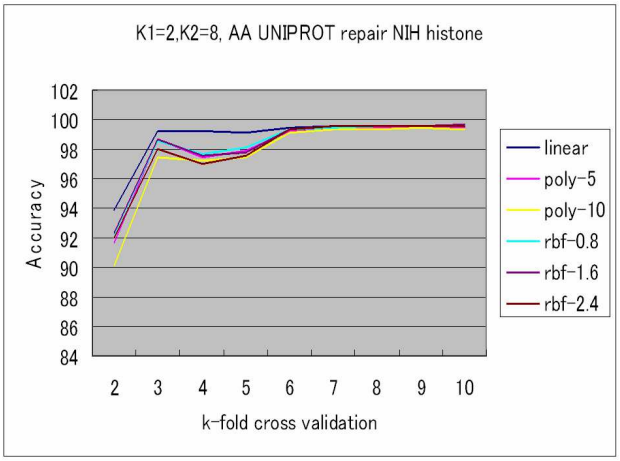


Figure 32: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and a set of nuclear non-repair proteins from Uniprot as negative data. K2=8 indicates that we have added over 6,000 dimensions of information measuring α-helix, β sheet, and other structural role sequence patterns.





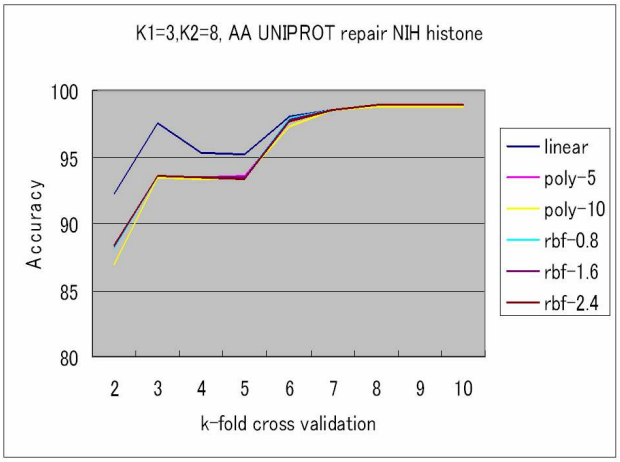


Figure 33: Results of 1-, 2-, and 3-spectrum amino acid primary sequence kernels using DNA repair proteins from Uniprot as positive data and histones from NIH as negative data. K2=8 indicates that we have added over 6,000 dimensions of information measuring α-helix, β sheet, and other structural role sequence patterns.

Table 8: The results of cross validation, given in accuracy (%), using an SVM with linear and polynomial kernels after the data was preprocessed using different spectrum kernel lengths. Size indicates the size of the dataset for a particular type of repair protein. Experiments were done using primary structure amino acid sequences and no secondary structure information.

Classification of KEGG DNA repair proteins

		1-spectrum		2-spectrum		
Repair Type	Size	5-fold	10-fold	5-fold	10-fold	Average
Excision	110	97.058	97.073	97.058	97.073	97.067
Mismatch	1054	71.834	71.832	88.761	89.783	80.553
Recombination	675	81.966	81.966	88.625	89.302	85.465
Helicase	89	97.624	97.622	97.624	97.622	97.623
Nuclease	137	96.342	96.343	96.314	96.316	96.329
Other	1677	63.959	65.043	74.927	96.684	70.153
Average		84.797	84.797	90.552	91.130	87.864

where there was also approximately a 10% performance boost after using the higher level spectrum kernels.

Results, shown in Table 8, suggest that there are still improvements to be made in the general classification case for proteins other than the five specific types isolated in these experiments. In Chapter 5, we will discuss a project to target this improvement.

4.4.2 The INTREPED Web Server

As we have achieved more than 80% accuracy in DNA repair protein prediction using only primary sequence spectrum kernels, it is useful to release this technique to the DNA repair community in a way that new proteins can be tested for DNA repair protein family membership. For this purpose, we have implemented a freely available web server called INTREPED, the INTeractive dna REPair prEDiction server, avaiable at *http://sunflower.kuicr.kyoto-u.ac.jp/~jbbrown/dnaRepairPrediction*.

Use of the web server is shown in Figure 34, where the user is presented a

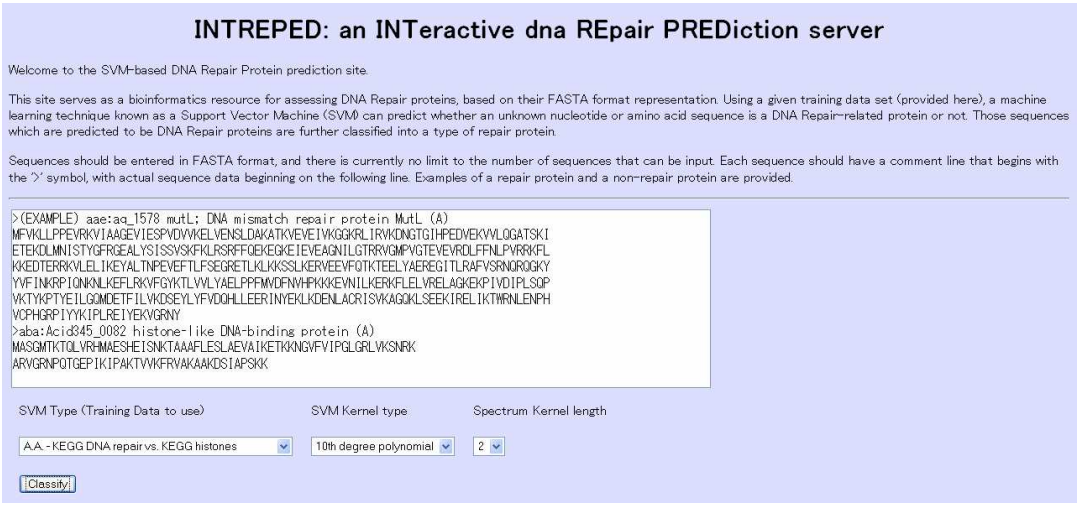
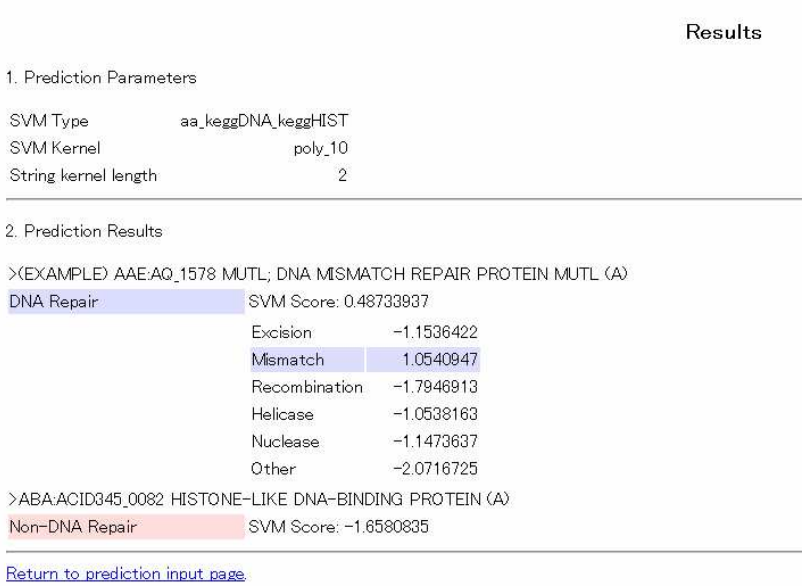


Figure 34: The input interface to the INTREPED web server. Users can select from different training sets, primary sequence spectrum kernel level, and similarity measure kernel to use in classification of an input protein. Proteins can be input as nucleotide or amino acid sequences.

prompt to input FASTA format sequences. They can also select which type of training data to use for their classification, in which the data corresponds to the data used in this thesis. For the purposes of speed and minimizing system load, we offer 1- and 2-spectrum primary structure sequence kernel processing.

After inputting the FASTA sequences, the web server performs recognition experiments on the input proteins and outputs whether or not they were detected to be related to DNA repair. For those proteins that are determined to be related to DNA repair, they are further classified into one of six types, similar to the classification experiment done in this chapter. Classification scores are output so that the user can visualize whether or not there was a clear and confident distinction in the classification process. Sample output from the web server is shown in Figure 35.

We believe that INTREPED is the first of its kind for DNA repair and has the potential to help advance knowledge of DNA repair by providing an interactive way for scientists in the field to quickly determine if a newly sequenced protein has a connection to DNA repair as well as what that protein's role might be. As of this writing, there are three principle datasets available for


Results

1. Prediction Parameters

SVM Type	aa_keggDNA_keggHIST
SVM Kernel	poly_10
String kernel length	2

2. Prediction Results

>(EXAMPLE) AAE:AQ_1578 MUTL; DNA MISMATCH REPAIR PROTEIN MUTL (A)

DNA Repair — SVM Score: 0.48733937

Excision	-1.1536422
Mismatch	1.0540947
Recombination	-1.7946913
Helicase	-1.0538163
Nuclease	-1.1473637
Other	-2.0716725

>ABA:ACID345_0082 HISTONE-LIKE DNA-BINDING PROTEIN (A)

Non-DNA Repair — SVM Score: -1.6580835

Return to prediction input page.

Figure 35: The prediction result page displayed after an input is processed by the INTREPED web server. Proteins that are recognized as DNA repair proteins are further classified into one of six categories. When classification occurs, the SVM scores resulting from classification are shown as well.

use as training data, though plans are in place to expand this to other datasets which are more comprehensive in range of function.

4.5 Conclusion

In this chapter, we have shown how machine learning techniques are able to successfully distinguish DNA repair-related proteins from non-repair-related proteins. We transformed our data via spectrum kernels into a representation that the Support Vector Machine could interpret, validated its performance using cross-validation, and experimented with both the use and non-use of secondary structure information. Overall, 80-99% accuracy was achieved in identifying DNA repair proteins, and some classes of DNA repair proteins could be classified with 95% or more accuracy.

We also identified the conditions for which polynomial and RBF kernels excel. This is an important piece of knowledge when considering how to evaluate a protein of unknown function. Whether computational resources are scarce or

abundant will impact how we choose to transform a protein under consideration into numerical data and learn from it.

The work here is an important step, albeit introductory, to automatically identifying DNA repair proteins in new genomes and furthering man's knowledge on the topic. By building gradually more complex predictive computational models for DNA repair, we may be able to simulate the repair process entirely in a computer, which would then lead to research on how to strengthen the repair system against failure.

A large-scale test on all of the currently sequenced and publicly available genomes would help to firmly grasp the advantages and disadvantages of the techniques presented here. We could then cross reference the result against expert knowledge databases or literature [20]. If we establish the possibility that there are more proteins computationally known than actually discovered in laboratories, we can provide an additional sense of direction for laboratory biologists in their experiments to actually observe DNA repair proteins.

We defer the discussion of projects that build on the accomplishments of this chapter to Chapter 5.

Chapter 5 Conclusion

Using a combination of algorithms on graphs and machine learning, we have demonstrated approaches to the problems of identification and structural prediction of DNA Repair proteins. Furthermore, we have created a public resource for the DNA Repair research community to identify new proteins that can be useful in extending our knowledge of immunology.

We see these two developments as simply the first step in building a more complete and useful DNA Repair Bioinformatics array of tools. Many extensions and new directions on this work are possible, and we elaborate on a number of these extensions here.

As discussed at the end of Chapter 3, there are many extensions that are possible for further work on the side chain packing problem. However, since the SCPP is fairly complete as a research topic, we find it to be more valuable to focus future work efforts on SVM-related and kernel research.

In addition to the SVM trained and used in this thesis that is useful as a general purpose DNA Repair protein identifier, SVMs for individual species would also be useful.

As the complete genomic sequence information of organisms rapidly increases, there is an increased need to automatically identify portions of genomes that could be related to DNA Repair. Using Hidden Markov Models, the DNA repair SVMs developed in this thesis, or other heuristic techniques, the computational identification of un-annotated sequences as DNA Repair proteins could be a considerable aide to laboratory testing and verification of new types of repair-related proteins.

The development of a customized kernel technique for machine learning (not necessarily limited to SVM; consider Kernel Principal Component Analysis) is a very high priority. Since a number of properties that exist in DNA Repair motifs are known, the opportunity has arisen to incorporate this knowledge into computational techniques. A specialized kernel that better identifies, and potentially classifies, types of repair proteins even in the presence of unrefined datasets is a valuable technique. It has been shown in this thesis that the unre-

fined KEGG DNA Repair dataset achieves 75-93% accuracy in discrimination, though parts of this dataset achieved only 60% accuracy in classification. Still, as these results were achieved using only the standard linear, polynomial, and RBF kernels, it remains possible to improve the accuracy using domain-specific knowledge to build a custom kernel method.

A completely different kernel (and ensuing SVM or other learning method) is possible for predicting whether a single or group of DNA Repair proteins that interacts with a DNA sequence repairs it or not. Using a machine learning approach would require either the accumulation of laboratory experiment information in order to train the machine to distinguish correct repair from incorrect repair, or a heuristic technique to make a decision in the absence of actual experiments. This approach can be further refined to deal with specific illnesses, and include other environmental parameters that can be quantized. Completion of such a technique would allow one to scan a patient suffering from a set of symptoms and determine if their DNA Repair system is functioning as expected.

The application of protein-protein interaction prediction techniques, using the kernels mentioned here as well as other research models proposed, is useful to try and uncover unknown DNA repair reactions. It is possible to then consider how DNA repair interaction networks fit into a more general PPI framework. The KEGG database has pathway information available to assist this.

Although initially more for theoretical purposes, researching the networking properties of DNA Repair proteins in specific organisms remains an open topic. In other words, we ask the question, "Does DNA Repair exhibit some well known network properties, such as scale-free connectivity, preferential attachment, or k-connectivity?" By gaining more insight into the network properties of DNA Repair, we may be able to better understand why it fails under certain conditions, leading to aging and disease. Through an understanding of network properties, we might also be able to better understand the damage tolerance mechanism which is one of the contributing factors to mutations and evolution.

Acknowledgments

The author is grateful first and foremost to his thesis advisor, Dr. Tatsuya Akutsu, for the constant support, discussion, and advice provided over the course of this thesis. Without Professor Akutsu's guidance, the work presented here would not have been possible.

Thanks are in order to Drs. Dukka Bahadur K.C., Morihiro Hayashida, Hiroto Saigo, and Nobuhisa Ueda, as well as to Mr. Lidio Meireless for their friendships and innumerable discussions which yielded insight and additional direction for this research.

The author wishes to particularly thank the Japanese Ministry of Education, Sports, Science, Culture, and Technology for selection to a graduate school study abroad program and the accompanying financial support. Computational resources to conduct and present research were provided by the Bioinformatics Center, Institute for Chemical Research, Kyoto University.

The utmost respect and appreciation is directed to my fiancé and her family for their unwavering support and encouragement, especially as I quietly kept to myself working for countless hours despite being a guest in their home.

Finally, I wish to thank my family for their unconditional support and love, even from afar, including non-relatives who have equally been my family for many years.

References

[1] T. Akutsu. Np-hardness results for protein side-chain packing. *Genome Informatics*, 8:180–186, 1997.

[2] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*. Garland Publishing, Inc., 19 Union Square West, New York, 10003, 1998.

[3] C.B. Anfinsen, E. Haber, M. Sela, and F.H. White Jr. Kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences of the United States of America*, 47:1309–1314, 1961.

[4] K.C.D. Bahadur. *Clique-based Algorithms for Protein Structure Prediction*. Ph.D. dissertation, Kyoto University, March 2006.

[5] K.C.D. Bahadur, J.B. Brown, E. Tomita, J. Suzuki, and T. Akutsu. Large scale protein side chain packing based on maximum edge-weight clique finding algorithm. *Proceedings of 2005 International Joint Conference of InCoB, AASBi and KSBI, BIOINFO 2005*, pages 228–233, 2005.

[6] K.C.D. Bahadur, E. Tomita, J. Suzuki, and T. Akutsu. Protein side-chain packing problem: A maximum edge-weight clique algorithmic approach. *Journal of Bioinformatics and Computational Biology*, 3:103–126, 2005.

[7] A. Bairoch, R. Apweiler, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O'Donovan, N. Redaschi, and L.S. Yeh. The universal protein resource (uniprot). *Nucleic Acids Research*, 33:D154–159, 2005.

[8] A-L. Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume, New York, New York, 2003.

[9] A. Bateman, L. Coin, R. Durbin, D. Finn, V. Hollich, S. Griffith-Jones, A. Khanna, M. Marshall, S. Moxon, E. Sonnhammer, D. Studholme, C. Yeats, and S. Eddy. The pfam protein families database. *Nucleic Acids Research*, 32:D138–141, 2004.

[10] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[11] M. Bhasin, E.L. Reinharz, and P. Reche. Recognition and classification of histones using support vector machine. *Journal of Computational Biology*, 13:102–112, 2006.

[12] J. Campbell and S. Fist. Cellular phones and cancer. Online, http://www.cqs.com/cell.htm.

[13] A. Canutescu and R.D. Shelenkov Jr. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12:2001–2014, 2001.

[14] J. Cheng, A. Randall, M. Sweredoski, and P. Baldi. Scratch: a protein structure and structural feature prediction server. *Nucleic Acids Research*, 33:w72–76, 2005.

[15] N. Christianini and J. Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, England, 2000.

[16] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[17] F. Crick and J. Watson. The complementary structure of deoxyribonucleic acid. In *Proceedings of the Royal Society of London*, number 1152 in Mathematical and Physical Sciences, pages 80–96, April 1954.

[18] R. Diestel. *Graph Theory, Third Edition*. Number 173 in Graduate Texts in Mathematics. Springer, Heidelberg, Berlin, 2006.

[19] O. Eriksoon, Y. Zhou, and A. Elofsson. Side chain-positioning as an integer programming problem. *Proceedings of the First International Workshop on Algorithms in Bioinformatics*, pages 128–141, 2001.

[20] E. Friedberg, G. Walker, W. Siede, R. Wood, R. Schultz, and T. Ellenberger. *DNA Repair and Mutagenesis*. ASM Press, Washington D.C., 2006.

[21] P.C. Hanawalt. Transcription-coupled repair and human disease. *Science*, 266:1957–1958, 1994.

[22] M. Hayashida, N. Ueda, and T. Akutsu. Inferring strengths of protein-

protein interactions from experimental data using linear programming. *Bioinformatics*, 19:ii58–65, 2003.

[23] T. Joachims. *Making large-scale support vector machine learning practical*, chapter 11, pages 169–184. MIT Press, 1999.

[24] M. Kanehisa, S. Goto, M. Hattori, K.F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res.*, 34:D354–357, 2006.

[25] J. Kimball. Dna repair. Online, http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/D/DNArepair.html, 2005.

[26] S. Lang. *Undergraduate Analysis*. Springer, New York, 1997.

[27] S. Linn. History of dna repair - life in the serendipitous lane: Excitement and gratification in studying dna repair. Online, http://videocast.nih.gov/ram/dnarig062006.ram, 2006.

[28] L. Mariño-Ramírez, B. Hsu, A. Baxevanis, and D. Landsman. The histone database: a comprehensive resource for histones and histone fold-containing proteins. *Proteins*, 62:838–842, 2006.

[29] N.J. Mulder, R. Apweiler, T.K. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bradley, P. Bork, P. Bucher, L. Cerutti, R. Copley E. Courcelle, U. Das, R. Durbin, W. Fleischmann, J. Gough, D. Haft, N. Harte, N. Hulo, D. Kahn, A. Kanapin, M. Krestyaninova, D. Lonsdale, R. Lopez, I. Letunic, M. Madera, J. Maslen, J. McDowall, A. Mitchell, A.N. Nikolskaya, S. Orchard, M. Pagni, C.P. Ponting, E. Quevillon, J. Selengut, C.J. Sigrist, V. Silventoinen, D.J. Studholme, R. Vaughan, and C.H. Wu. Interpro, progress and status in 2005. *Nucleic Acids Research*, 33:D201–205, 2005.

[30] National Institutes of Health DNA Repair Special Interest Group. What is dna repair? Online, http://www.nih.gov/sigs/dna-rep/whatis.html, 2005.

[31] R. Samudrala and J. Moult. An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. *Journal of Molecular Biology*, 257:893–914, 1998.

[32] A. Sancar. Mechanisms of dna excision repair. *Science*, 266:1954–1956,

1994.

[33] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, England, 2004.

[34] J. Suzuki, E. Tomita, and T. Seki. An algorithm for finding a maximum clique with maximum edge-weight and computational experiments. *The Information Processing Society of Japan: Technical Report MPS*, pages 45–48, 2002.

[35] E. Tomita and T. Seki. An efficient branch-and-bound algorithm for finding a maximum clique. *Proceedings of DMTCS 2003, LNCS*, 2731:278–289, 2003.

[36] R.J. Wilson. *Introduction to Graph Theory, Fourth Edition*. Prentice Hall, Edinburgh Gate, Harlow, Essex, England, 1996.

[37] Z. Xiang and B. Honig. Extending the accuracy limits of prediction for side-chain conformations. *Journal of Molecular Biology*, 311:421–430, 2001.

[38] J. Xu. Rapid protein side-chain packing via tree decomposition. *9th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005)*, pages 423–429, 2005.

List of Publications by the Author

Conference Proceedings

1. J.B. Brown, Dukka Bahadur K.C., Etsuji Tomita, Tatsuya Akutsu. Multiple methods for protein side-chain packing using maximum weight cliques. *Genome Informatics Series*, Vol. 17, No. 1, 3–12, 2006
2. Dukka Bahadur K.C., J.B. Brown, Etsuji Tomita, Jun'ichi Suzuki, Tatsuya Akutsu. Large scale protein side-chain packing based on maximum edge-weight clique finding algorithm. *Proceedings of the 2005 International Joint Conference of InCoB, AASBi, and KSBI*, 228–233, 2005

Poster Presentations

1. J.B. Brown, Tatsuya Akutsu. DNA repair recognition via support vector machines. *The Seventeenth International Conference on Genome Informatics 2006*, Poster No. 108, 2006.
2. Dukka Bahadur K.C., J.B. Brown, Etsuji Tomita, Jun'ichi Suzuki, Tatsuya Akutsu. Large scale protein side-chain packing based on maximum edge-weight clique finding algorithms. *The Sixteenth International Conference on Genome Informatics 2005*, Poster No. 128, 2005.